UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DIRETORIA DE ENSINO E PESQUISA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
CURSO DE ENGENHARIA INDUSTRIAL ELÉTRICA – ÊNFASE ELETROTÉCNICA

WESLEY JOSÉ DOS SANTOS FURLANETTO

DESENVOLVIMENTO DE PROTÓTIPO DE UM DISPOSITIVO “BABÁ ELETRÔNICA” PARA PESSOAS COM DEFICIÊNCIA AUDITIVA

TRABALHO DE CONCLUSÃO DE CURSO

CORNÉLIO PROCÓPIO
2019

WESLEY JOSÉ DOS SANTOS FURLANETTO

DESENVOLVIMENTO DE PROTÓTIPO DE UM DISPOSITIVO “BABÁ ELETRÔNICA” PARA PESSOAS COM DEFICIÊNCIA AUDITIVA

Trabalho de Conclusão de Curso de graduação, do Curso de Engenharia Industrial Elétrica – Ênfase Eletrotécnica da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de Engenheiro.

Orientador: Prof. Dr. Luiz Marcelo Chiesse da Silva.
Coorientador: Prof. Dr. Wagner Endo.

CORNÉLIO PROCÓPIO

2019

Universidade Tecnológica Federal do Paraná Campus
Cornélio Procópio
Departamento Acadêmico de Elétrica
Curso de Engenharia Industrial Elétrica

FOLHA DE APROVAÇÃO

Wesley José dos Santos Furlanetto

Desenvolvimento de protótipo de um dispositivo "babá eletrônica" para pessoas com deficiência auditiva.

Trabalho de conclusão de curso apresentado às 14:40hs do dia 19/06/2019 como requisito parcial para a obtenção do título de Engenheiro Eletricista no programa de Graduação em Engenharia Industrial Elétrica da Universidade Tecnológica Federal do Paraná. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Prof. Dr. Luiz Marcelo Chiesse da Silva - Presidente (Orientador)

_____
Prof. Dr. Wagner Endo - (Coorientador)

_____
Prof. Dr. Paulo Rogério Scalassara - (Membro)

_____
Prof. Me. Marco Antonio Ferreira Finocchio - (Membro)

A folha de aprovação assinada encontra-se na coordenação do curso.

Aos meus pais, Rubens e Leonice, que deram todo o apoio e incentivo para que eu pudesse realizar todos os meus sonhos e também por todos os exemplos e contraexemplos de vida.

Aos meus padrinhos e a outros familiares que me ajudaram a conseguir chegar até aqui.

À minha irmã, Isabelle, que serviu de inspiração para o desenvolvimento deste trabalho e à minha outra irmã, Vivien, que sempre me apoiou em todos os momentos.

Às memórias de Maria Augusta e Giovanna, avó e irmã que me ensinaram o valor da simplicidade da vida.

Especialmente ao meu filho, Davi, que tenha o mundo cheio de alegrias, sabedoria e vitórias.

AGRADECIMENTOS

Primeiramente, agradeço a Deus por toda a minha vida e por permitir que chegasse até a esta etapa importante.

Agradeço a Universidade Tecnológica Federal do Paraná (UTFPR), seu corpo docente, direção e administração, que permitiram uma sólida formação profissional e ética.

Agradeço aos professores orientadores Dr. Luiz Marcelo Chiesse da Silva e Dr. Wagner Endo, pela oportunidade e apoio à elaboração deste trabalho e também aos demais professores que contribuíram com a minha formação acadêmica.

Agradeço aos meus pais e familiares, pelo suporte dado ao longo desta graduação.

Agradeço aos amigos que souberam compreender a distância causada nesta etapa, mas que mesmo assim estiveram presentes e também aos amigos que fiz ao longo destes anos de graduação, que certamente estarão presentes por toda a minha vida.

A todos que contribuíram diretamente e indiretamente na minha formação, registro o meu muito obrigado.

The progressive development of man is vitally dependent on invention. It is the most important product of his creative brain. Its ultimate purpose is the complete mastery of mind over the material world, the harnessing of the forces of nature to human needs. (TESLA, Nikola, 1919, p. 696).

O desenvolvimento humano depende fundamentalmente da invenção. Ela é o produto mais importante de seu cérebro criativo. Seu objetivo final é o completo domínio da mente sobre o mundo material e o aproveitamento das forças da natureza em favor das necessidades humanas. (TESLA, Nikola 1919, p. 696).

RESUMO

FURLANETTO, Wesley J. S. Desenvolvimento de protótipo de um dispositivo “babá eletrônica” para pessoas com deficiência auditiva. 2019. 83 f. Trabalho de Conclusão de Curso (Engenharia Industrial Elétrica – Ênfase Eletrotécnica) – Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2019.

Este trabalho tem por objetivo a construção de um protótipo de baixo custo de um dispositivo do tipo “babá eletrônica” para pessoas com deficiência auditiva que são responsáveis pelos cuidados de bebês. É dever da sociedade criar meios que façam com que as condições de educação e trabalho, e de vida cultural e social sejam feitas acessíveis a todas as pessoas com deficiência e para que estas também consigam a liberdade de aprender com suas próprias escolhas na educação de seus dependentes. Este protótipo atende os conceitos de equiparação de oportunidades e vida independente, além de ser funcional para pessoas sem deficiência, o caracterizando como um dispositivo que possui um desenho universal. Utilizando a plataforma de desenvolvimento de *software Arduino* em conjunto com um módulo de conexão *WiFi*, este protótipo é classificado como um dispositivo do tipo Internet das Coisas pois integra comunicação entre dispositivos conectados na rede. Este projeto inclui o desenvolvimento de um aplicativo compatível com *smartphones* e *tablets* que fazem uso do sistema operacional *Android*, que serão responsáveis pelo sistema de alerta aos usuários.

Palavras-chave: Tecnologia assistiva. Sistemas embarcados. Internet das Coisas.

ABSTRACT

FURLANETTO, Wesley J. S. Desenvolvimento de protótipo de um dispositivo “babá eletrônica” para pessoas com deficiência auditiva. 2019. 83 f. Trabalho de Conclusão de Curso (Engenharia Industrial Elétrica – Ênfase Eletrotécnica) – Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2019.

This work aims to build a low-cost prototype of an "electronic babysitter" device for hearing impaired people who are responsible for the care of infants. It is the duty of society to create means to ensure that conditions of education and work, cultural and social life are made accessible to all people with disabilities and these also able freedom to learn from their own choices in the education of their dependents. This prototype meets the concepts of equalization of opportunities and independent living, in addition to being functional for people without disabilities, characterizing it as a device that has a universal design. Using the Arduino software development platform in conjunction with a Wi-Fi connection module, this prototype is classified as an Internet of Things device due to the communication between devices connected to the network. This project includes developing an app compatible with smartphones and tablets that make use of the Android operational system, because these mobile devices will be responsible for the warning system to users.

Keywords: Assistive technology. Embedded systems. Internet of Things.

LISTA DE FIGURAS

LISTA DE FOTOGRAFIAS

LISTA DE QUADROS

SIGLAS

ADC	Conversão analógico para digital
ALU	Unidade Lógica Aritmética
AP	*Access Point*
API	Interface de Programação de Aplicativos
CPU	Unidade de Processamento Central
CSS	*Cascading Style Sheets*
CU	Unidade de Controle
DTLS	*Datagram Transport Layer Security*
F0	Frequência fundamental
F1	Primeiro formante
F2	Segundo formante
FBI	*The Federal Bureau of Investigation*
FFT	Transformada Rápida de Fourier
GND	*Ground*
GPIO	*General Purpose Input Output*
GPS	*Global Position System*
HTML	*HyperText Markup Language*
HTTP	*Hyper Text Transfer Protocol*
HTTPS	*Hyper Text Transfer Protocol Secure*
I/O	*Input/Output*
I2C	*Inter-Integrated Circuit*
I2S	*Inter-IC Sound*
IDE	Ambiente de Desenvolvimento de *Software*
IOS	Sistema operacional *iPhone*
IP	*Internet Protocol*
LCD	Display de Cristal Líquido
LSB	*Less Significant Bit*
M2M	Máquina-para-máquina
MQTT	*Message Queuing Telemetry Transport*
PWM	*Pulse With Modulation*
RISC	Computador com um conjunto reduzido de instruções
RST	*Reset*
RX	Sinal de Recepção Serial

SCL	*Serial Clock*
SDA	*Serial Data*
SDH/PR	Secretaria de Direitos Humanos da Presidência da República
SDK	Kit de Desenvolvimento de *Software*
SNPD	Secretaria Nacional de Promoção de Direitos da Pessoa com Deficiência
SPI	*Serial Peripheral Interface*
SRAM	*Static Random Access Memory*
STA	*Station Mode*
TCP	*Transmission Control Protocol*
TLS	*Transport Layer Security*
TX	Sinal de transmissão serial
UDP	*User Datagram Protocol*
URL	*Uniform Resource Locator*
USB	*Universal Serial Bus*
VCC	Tensão de alimentação contínua
XMPP	*Extensible Messaging and Presence Protocol*

ABREVIATURAS

App	Aplicativo
BaaS	*Backend as a Service*
CoAP	*Constrained Application Protocol*
µC	Microcontrolador

ACRÔNIMOS

IoT	Internet das Coisas
JSON	*JavaScript Object Notation*
RAM	*Random Access Memory*
REST	Transferência de Estado Representacional
ROM	*Read Only Memory*
UART	*Universal Asynchronous Receiver/Transmitter*

SUMÁRIO

1 INTRODUÇÃO

Os dispositivos conhecidos como “babás eletrônicas” surgiram da necessidade da monitoração de crianças por cuidadores - pessoas responsáveis pela supervisão e cuidados - que não poderiam estar no mesmo ambiente em que a criança se encontrava. O primeiro dispositivo é datado de 1938, idealizado por Eugene F. McDonald Jr., presidente da *Zenith Radio Corporation*. Preocupado devido ao sequestro de Charles Lindbergh Jr., que ocorreu em 1932 (FEDERAL BUREAU OF INESTIGATION, 2008), decidiu criar um sistema pontual de microfones e receptores que monitorasse o quarto de sua filha. O transmissor foi chamado de “*Guardian Ear*” e o receptor foi nomeado como “*Radio Nurse*” (GREUTHER, 2014).

Devido ao avanço tecnológico, melhorias foram implementadas nestes dispositivos. Desde a transmissão do som, que deixou de ser realizada através da rede de energia, pois esta causa interferência com o surgimento de ruídos, até a redução do tamanho dos dispositivos.

Atualmente, existem modelos que não utilizam apenas o som como variável de monitoramento, há dispositivos que englobam captura de imagens e termômetros, enviando os dados para *smartphones* e *tablets*, entretanto o custo de aquisição deste aparelho é relativamente elevado.

Também estão disponíveis para *download*, aplicativos (*Apps*) gratuitos para celulares e *tablets*, que exercem a função de babá eletrônica, porém, este recurso necessita da comunicação entre dois celulares ou *tablets*, restringindo o uso em um dos dois aparelhos e os sinais de alerta são enviados para um único dispositivo.

Devido à inclusão social de pessoas com deficiência, equipamentos e dispositivos como “babás eletrônicas” devem ser adaptados de forma a atender este grupo de pessoas, que no Brasil é composto por cerca de 45,6 milhões de pessoas (23,9% da população brasileira) de acordo com a cartilha do censo 2010 (SECRETARIA DE DIREITOS HUMANOS DA PRESIDÊNCIA DA REPÚBLICA; SECRETARIA NACIONAL DE PROMOÇÃO DE DIREITOS DA PESSOA COM DEFICIÊNCIA; COORDENAÇÃO-GERAL DO SISTEMA DE INFORMAÇÕES SOBRE A PESSOA COM DEFICIÊNCIA, 2012, p. 6). Deste modo, estes tipos de equipamentos, principalmente os de supervisão e segurança física, devem possibilitar sua utilização por pessoas com deficiência auditiva, que representam cerca de 5,1% dos brasileiros (SECRETARIA DE DIREITOS HUMANOS DA PRESIDÊNCIA DA

REPÚBLICA; SECRETARIA NACIONAL DE PROMOÇÃO DE DIREITOS DA PESSOA COM DEFICIÊNCIA; COORDENAÇÃO-GERAL DO SISTEMA DE INFORMAÇÕES SOBRE A PESSOA COM DEFICIÊNCIA, 2012, p. 6).

Há trabalhos que também consistem na detecção do choro de uma criança, utilizando *Arduino*. Um exemplo é o trabalho “Babá eletrônica adaptada para surdos”. (FREITAS, 2010). Neste trabalho, o autor utiliza modelagem de reconhecimento através de análise espectral, algoritmo Transformada Rápida de Fourier (*FFT)*, Teoria dos conjuntos aproximados, uma ferramenta de análise de dados e um sistema especialista. Segundo Freitas (2010, p. 5), este projeto alcançou uma margem de acerto de 80% dos sons analisados.

Nesta seção serão apresentados os seguintes tópicos: descrição do problema, justificativa, objetivos, contribuições e a organização deste trabalho.

1.1 DESCRIÇÃO DO PROBLEMA

Um ouvinte consegue identificar, diferenciar e interpretar sons, como por exemplo: se uma sirene está ligada ou não, perceber se a intensidade sonora é elevada ou baixa, entre outros. Para uma pessoa com deficiência auditiva realizar estas tarefas é necessária a existência de intermediadores - que podem ser pessoas ouvintes ou dispositivos eletrônicos de acessibilidade.

Como exemplo, quando um casal, ou um dos cônjuges, possui deficiência auditiva e deve cuidar de filhos recém-nascidos. Existe a dificuldade nos cuidados do bebê para verificar o momento em que esta criança despertou do sono ou se encontra chorando, quando não há ninguém próximo para observar a criança.

O problema abordado neste trabalho trata de uma forma de auxílio às pessoas com deficiência auditiva cuidadoras de crianças pequenas, como bebês, com a finalidade de promover o conceito de equiparação de oportunidades e de vida independente. Como solução, total ou parcial para esta situação, o presente projeto foi desenvolvido: a criação do protótipo de um dispositivo para a comunicação com aparelhos portáteis, alertando os responsáveis pela criança do choro da mesma, indicando esta ocorrência através de atuadores, tais como sinais luminosos e dispositivos de vibração.

Este dispositivo pode ser dividido, portanto, em duas partes principais: a primeira consiste em um sistema embarcado, responsável pela captura de áudio,

filtragem de sinais, conversão de sinais analógicos em digitais e transmissão de sinal discreto via rede sem fio; e a segunda parte consiste na elaboração de um *App* destinado a aparelhos portáteis móveis que possuem compatibilidade com sistemas operacionais *Android*, para a recepção deste sinal discreto e acionamento das funções de vibração e iluminação do aparelho móvel. O projeto proposto também envolve a criação de um programa responsável pela configuração da "babá eletrônica" pelo usuário do sistema, conexão do dispositivo à rede *wireless*, e alteração da amplitude sonora, abrangendo assim, as características dos sinais de áudio provenientes do choro de uma criança.

1.2 JUSTIFICATIVA

Existem dois fatores citados anteriormente que são a base deste trabalho, e os mesmos serão aprofundados como justificativa deste projeto. Tratam-se da equiparação de oportunidades e da vida independente de pessoas com deficiência.

Equiparação de oportunidades é descrito como:

> O processo mediante o qual os sistemas gerais da sociedade, tais como o meio físico, a habitação e o transporte, os serviços sociais e de saúde, as oportunidades de educação e trabalho, e a vida cultural e social, incluídas as instalações esportivas e de recreação, são feitos acessíveis para todos. Isto inclui a remoção das barreiras que impedem a plena participação das pessoas deficientes em todas estas áreas, permitindo-lhes assim alcançar uma qualidade de vida igual à de outras pessoas. (DRIEDGER; ENNS, 1987 apud SASSAKI, 2006, p. 38).

É necessário também o desenvolvimento do conceito de vida independente, que significa:

> Ter oportunidades para tomar decisões que afetam a própria vida, realizar atividades de própria escolha. (...) Vida independente tem a ver com a autodeterminação. E com o direito e a oportunidade para seguir um determinado caminho. E significa ter a liberdade de falhar e aprender das próprias falhas, tal qual fazem as pessoas não-deficientes. (IRLU,1990 apud SASSAKI, 2006, p. 50).

Portanto, é dever da sociedade desenvolver técnicas que abrangem a inclusão de pessoas com deficiência.

O projeto proposto neste trabalho possibilitará ao autor e a futuros alunos a abrangência de parte do conteúdo das áreas de controle, sistemas de comunicação, transmissão de dados, programação, gerenciamento de projetos, e possibilitará também uma aplicação do conhecimento adquirido no meio acadêmico com

problemas cotidianos, através da melhoria da qualidade de vida de pessoas com deficiência auditiva.

A principal motivação para este trabalho consiste em proporcionar uma ferramenta de auxílio a pessoas com deficiência auditiva a conseguirem independência para a tomada de decisões, quando forem solicitados, utilizando um desenho universal, que, segundo Sassaki, (2006, p.146-147) é mais vantajoso que um produto voltado apenas para pessoas com deficiência porque também atende as necessidades de pessoas sem deficiência.

1.3 OBJETIVOS

Este projeto consiste no desenvolvimento de um sistema embarcado “babá eletrônica”, de baixo custo, com a função de detectar e identificar sinais de áudio similares ao choro de um bebê, para a transmissão de um sinal discreto de alarme, via rede *wireless* (sem fio) para dispositivos portáteis compatíveis com a plataforma *Android*.

A seguir estão descritos quais são os objetivos desta proposta de trabalho de conclusão de curso.

1.3.1 Objetivo Geral

Desenvolvimento do protótipo de um dispositivo eletrônico remoto tipo “babá eletrônica” que atuará como um sistema de alerta para responsáveis por crianças, incluindo usuários com deficiência auditiva.

Este dispositivo deve transmitir um sinal de dados, a partir da identificação do sinal sonoro proveniente do choro de um bebê, através de comunicação *wireless*, para aparelhos portáteis móveis, como *smartphones* ou *tablet*s compatíveis com o sistema operacional *Android*.

1.3.2 Objetivos Específicos

Este projeto foi dividido em seis etapas, descritas a seguir:

a) obtenção de amostras de choros de bebês;
b) tratamento e conversão do sinal analógico;

c) análise e interpretação dos dados obtidos, utilizando um microcontrolador presente na placa de *hardware ESP8266 ESP-01*;
d) reconhecimento e classificação do sinal sonoro proveniente do choro da criança;
e) transmissão do sinal de dados via rede sem fio;
f) desenvolvimento do *App* para sistemas *Android*, capaz de receber o sinal e acionar mecanismos presentes no aparelho móvel, que funcionarão como atuadores do sistema de alarme.

Ao fim de todas as etapas, o projeto e aquisição de dados foram finalizados, bem como a realização da análise dos mesmos para que sejam discutidas possíveis melhorias e ajustes do sistema de acordo com os resultados obtidos.

1.4 CONTRIBUIÇÕES

Este trabalho visa contribuir nas áreas de cuidados de crianças, podendo ser utilizado por profissionais da área de saúde em enfermarias e alas neonatais, por pedagogos e cuidadores e especialmente por pessoas portadoras de alguma deficiência auditiva, que sejam responsáveis pelos cuidados de um bebê.

1.5 ESTRUTURA DO TRABALHO

No capítulo 2 são apresentadas as fundamentações teóricas abordadas neste trabalho. No capítulo 3, estão dispostos quais os materiais e a metodologia utilizados para a elaboração do protótipo. O capítulo 4 discorre sobre como cada etapa foi realizada para a construção do protótipo tratado neste trabalho. No capítulo 5, serão apresentados e discutidos os resultados obtidos. O capítulo 6 trata das considerações finais, bem como possíveis melhorias para eventuais trabalhos futuros. Em seguida, estão dispostas as referências utilizadas, os apêndices e os anexos inerentes ao presente trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

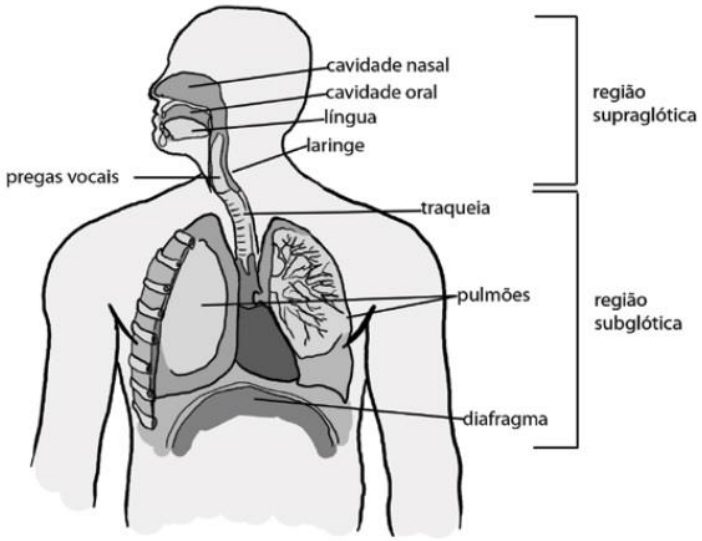
Neste capítulo, serão apresentados os principais conceitos teóricos relacionados com a elaboração deste trabalho.

2.1 CARACTERIZAÇÃO DO CHORO

O choro de uma criança recém-nascida é a sua primeira atitude instintiva, utilizada como um sistema de alerta, indicando que a criança necessita de cuidados, como: alimentação, atenção ou enfermidade.

O choro é compreendido pela alternância rítmica entre o som produzido pelas pregas vocais, localizadas na laringe, na fase de expiração do sistema respiratório e pausas causadas pela fase de inspiração. A Figura 1, apresenta o aparelho fonador humano.

Figura 1 – Aparelho fonador humano

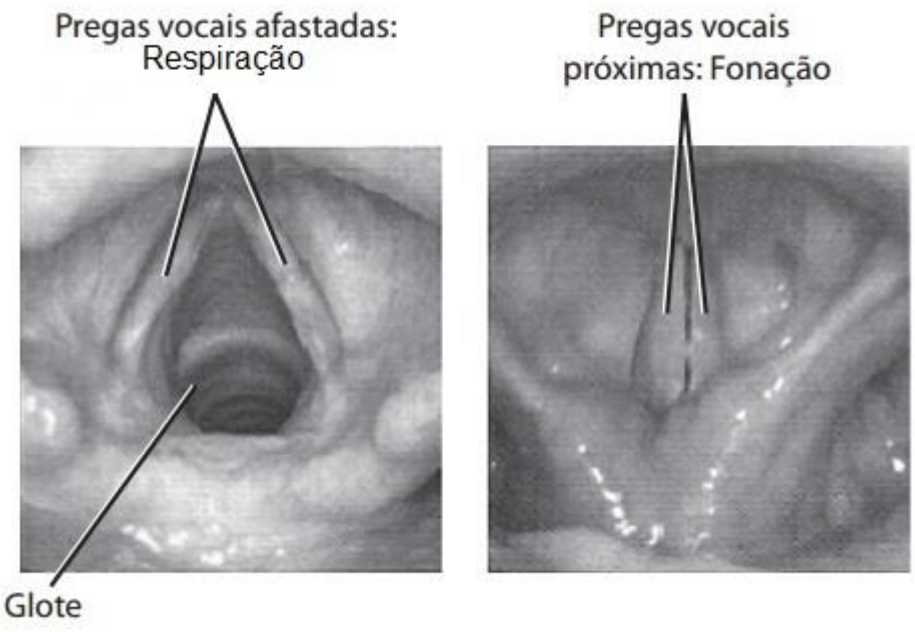


Fonte: Adaptado de Parker (2007).

A glote é o espaço entre as pregas vocais (SEARA; NUNES; LAZAROTTO-VOLCÃO, 2011, p. 17), funcionando como uma válvula: quando se encontra totalmente aberta, permite a passagem de ar sem obstrução, caracterizando a respiração. Ao fechar, é semelhante a um tubo de Venturi, aumentando a velocidade do ar e reduzindo a pressão no local da constrição (ÇENGEL; CIMBALA, 2007, p.318),

provocando a vibração das pregas vocais. A Figura 2 ilustra uma glote aberta e fechada.

Figura 2 – Abertura e fechamento da glote, na produção de sons



Fonte: Adaptado de Seara; Nunes; Lazarotto-Volcão (2011).

O número de vibrações por segundo das pregas vocais caracteriza a frequência fundamental (F0), que em recém-nascidos saudáveis, varia aproximadamente entre 250 a 450 Hz. (LAGASSE; NEAL; LESTER, 2005). A região subglótica está associada ao sistema nervoso autônomo e controla a intensidade do som e ritmo, incluindo a inalação e pausas de respiração. O som é modificado pelo tamanho e contorno do trato vocal superior, produzindo frequências de ressonância, chamadas de formantes, estão em uma faixa de frequência acima da F0. LaGasse, Neal e Lester (2005, p. 84) afirmam que os formantes mais utilizados para avaliação são o primeiro (F1) e o segundo (F2) e ocorrem aproximadamente em 1100 Hz e 3300Hz, respectivamente.

Basicamente, há três modos de choros identificáveis: fonação (F0), hiperfonação (1000 – 2000 Hz) e turbulento (vibração dissonante das pregas vocais) (SINGER; ZESKIND, 2001, p. 152).

2.2 PROCESSAMENTO DE SINAIS

Um sinal é um fenômeno variável que pode ser mensurado (WEEKS, 2012, p. 11). Na natureza, os sinais são em sua maioria contínuos, mas os sistemas computacionais trabalham com sinais discretos. Portanto, o sinal objeto de análise deste projeto deve ser transformado da forma analógica para a forma digital.

O microcontrolador presente na placa ESP8266 ESP-01 realiza a conversão de sinais analógicos em sinais discretos (digitais) com a finalidade de realizar a leitura de tensão interna. Logo, se faz necessária a utilização de um conversor analógico digital (ADC) externo. Com isso será possível a implementação de um *software* para a avaliação da ocorrência do choro de uma criança.

Weeks (2012, p. 17), afirma que um sistema realiza uma operação matemática em um sinal e pode ser representado por diagrama de blocos, como apresentado na Figura 3:

Figura 3 – Diagrama de bloco de um sistema

Fonte: Autoria própria (2016).

2.3 MICROCONTROLADORES

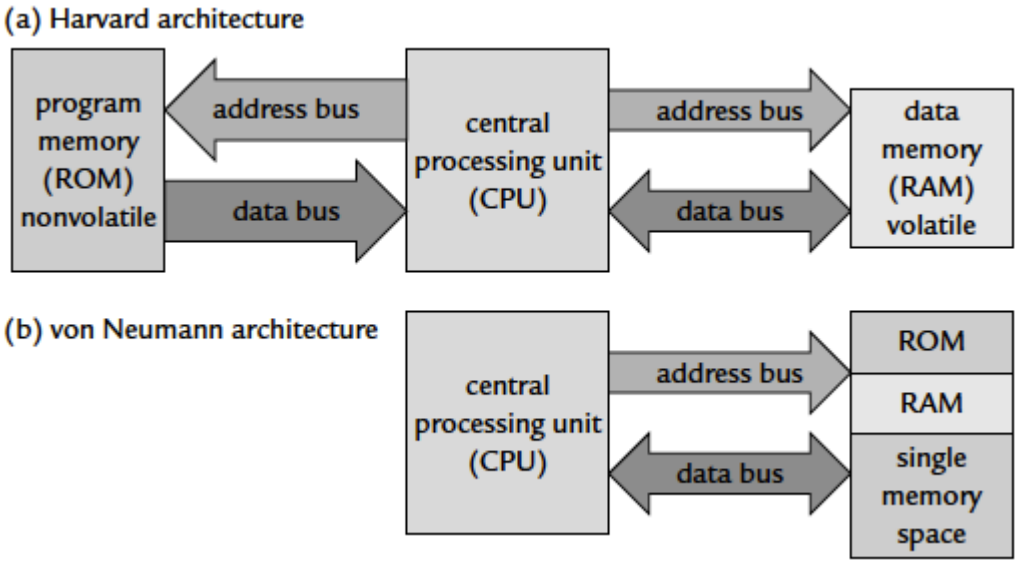
Microcontroladores (µC), segundo (PENIDO; TRINDADE, 2013), possuem internamente elementos como uma unidade central de processamento (*CPU – Central Processor Unit*), memórias, conjunto de entradas/saídas (*Input/Output – I/O*), e um conjunto de dispositivos auxiliares ao funcionamento.

2.3.1 Arquiteturas *Von Neumann* e *Harvard*

Os µC atuais são baseados em dois tipos de arquitetura principais: *von Neumann* e *Harvard*. Os µC es que utilizam uma única área de memória para alocar *software* e dados, conforme ilustra a Figura 4(b), se baseiam na arquitetura de *von*

Neumann. Esta arquitetura priva o acesso simultâneo de dados e programas, aumentando, portanto, o tempo de processamento. Na arquitetura *Harvard*, ilustrada pela Figura 4(a), é possível o acesso de dados e programas simultaneamente devido à locais distintos para os dois tipos de memória: não-volátil (*Read-Only Memory* – ROM), responsável pelo armazenamento do programa e volátil (*Random Access Memory* – *RAM*), responsável pela leitura e armazenamento de dados. Logo, por ter acesso às duas memórias simultaneamente, esta arquitetura trabalha com um processamento mais veloz se comparado à arquitetura *von Neumann*, para a mesma frequência de *clock.*

Figura 4 – Arquiteturas dos µC (a) Harvard (b) von Neumann



Fonte: Zanco (2010).

2.3.2 Unidade Central de Processamento

A *CPU*, é o principal elemento presente em um µC, cuja finalidade é o processamento de todos os tipos de dados, bem como a apresentação dos resultados destes processamentos. Sua principal função é garantir a execução correta do programa através do acesso às memórias, execução de cálculos, controle de periféricos e interrupções. (MICROCHIP, 2018).

A *CPU* pode ser dividida em três componentes principais:

a) unidade lógica aritmética (*ALU – Arithmetic Logic Unit*):

- responsável pelas operações matemáticas e lógicas do sistema microcontrolado;

b) unidade de controle (*Control Unit – CU*):

- extrai, decodifica e executa as instruções de memória, requisitando a *ALU* quando necessário;

c) registradores e memórias:

- encarregados do armazenamento de dados para o processamento.

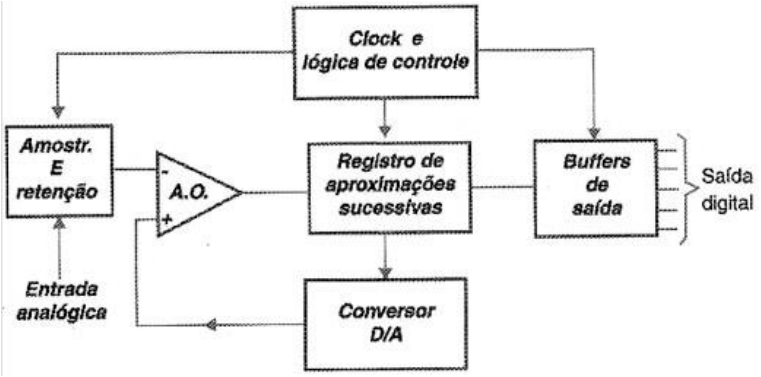
2.4 CONVERSÃO ANALÓGICA PARA DIGITAL

As grandezas físicas, como temperatura e pressão, são representadas geralmente por valores analógicos. Entretanto, os dados e informações utilizadas em um µC são variáveis discretas, sendo necessário, portanto, uma conversão dos valores analógicos para valores digitais.

Existem vários tipos de ADC tais como: comparador paralelo, rampa tipo contador, rampa dupla tipo integrador, aproximações sucessivas, dentre outros. Todos estes conversores realizam a mesma função, a de converter um sinal de forma analógica para digital, porém utilizando métodos diferentes.

A técnica de conversão utilizada neste trabalho é através de sucessivas aproximações. A Figura 5 indica um diagrama de blocos que representa este tipo de conversão.

Figura 5 – Diagrama de blocos de um ADC por aproximações sucessivas



Fonte: Braga (2010).

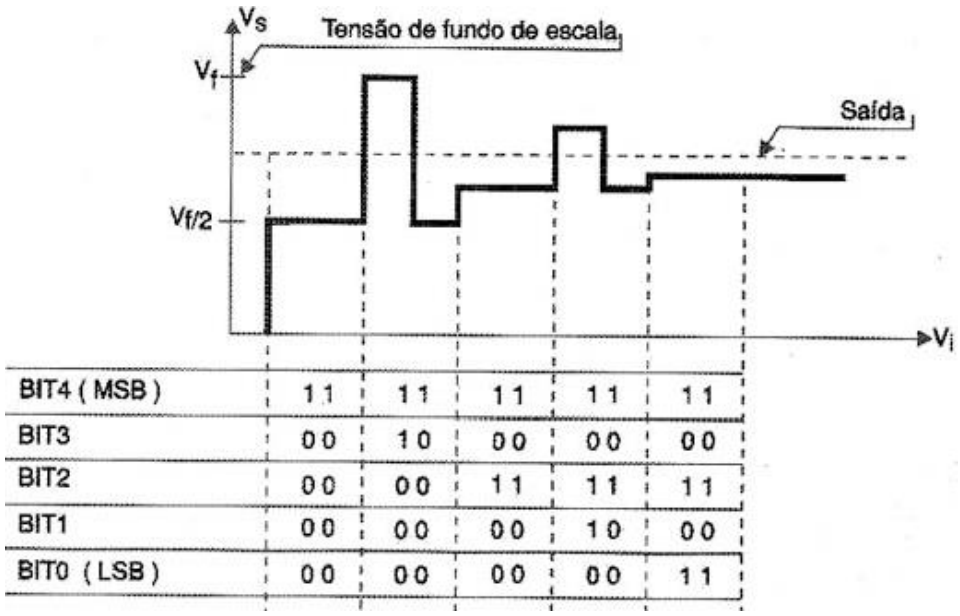
O sinal aplicado a entrada é retido pelo circuito de amostragem e retenção, aplicado à entrada do comparador e ao mesmo tempo dispara o circuito de *clock* do

setor de conversão digital. Ao iniciar a conversão o registrador de aproximações sucessivas começa colocando a “1” o *bit* mais significativo (*Most Significant Bit – MSB*) da saída, aplicando este sinal no ADC. Se com este procedimento, a tensão aplicada pelo ADC à entrada de referência do comparador for maior que a de entrada, isso demonstra que o valor que este *bit* representa é maior que o que se deseja converter.

O comparador informa isso ao registrador de aproximações, que então volta o *MSB* a zero e coloca o *bit* que o segue imediatamente a “1”. Uma nova comparação é feita. Se agora o valor da tensão for menor que a de entrada, este *bit* é mantido, e testa-se o seguinte, colocando a “1”. Se novamente o valor for ultrapassado, o comparador informa isso ao registro e o *bit* volta a zero passando o seguinte a “1” que é testado.

Quando todos os *bits* forem testados, haverá na saída do registro um valor binário muito próximo do desejado, dependendo da resolução do circuito. Testando todos os *bits* desta forma, a conversão se torna muito rápida, já que não será preciso esperar a contagem até o final, conforme mostra o gráfico ilustrado pela Figura 6:

Figura 6 – Exemplo de ADC utilizando o método de aproximações sucessivas



Fonte: Braga (2010).

Após a finalização da conversão de uma amostra, o resultado encontrado no registrador ADC é dado pela Equação (1):

$$ADC = \frac{V_{IN} \cdot \mathrm{Res}}{V_{REF}} \quad (1)$$

Onde:

V_{IN} é a tensão sobre o pino de entrada selecionado;

V_{REF} é a tensão de referência utilizada.

Res é a resolução do conversor.

Para encontrar o valor de Res, utiliza-se a Equação (2):

$$Res = 2^b \quad (2)$$

Onde b é a quantidade de *bits* da resolução do conversor.

2.5 PROTOCOLO DE COMUNICAÇÃO *I2C*

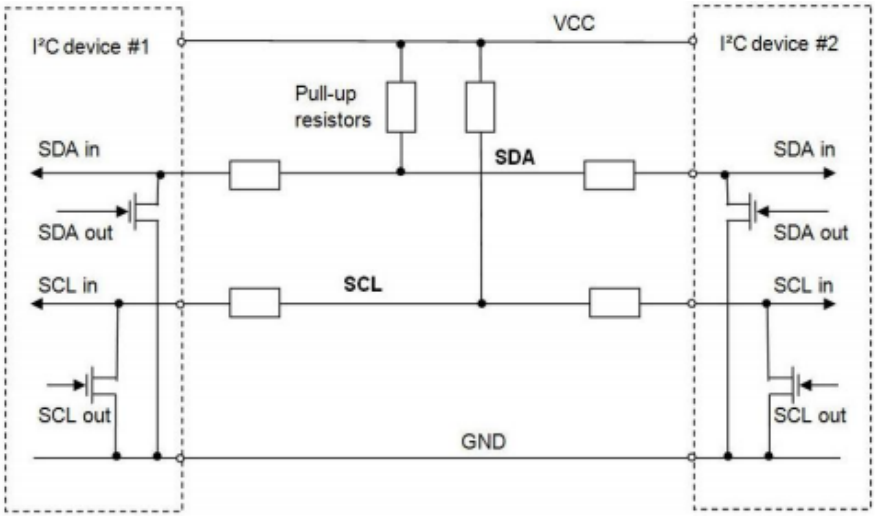
O protocolo *Inter IC Communication* (*I2C*), desenvolvido pela empresa *Philips* em 1982, tem como principal objetivo simplificar a comunicação entre uma *CU* e diversos dispositivos de baixa complexidade pertencentes ao mesmo circuito (KALINSKY; KALINSKY, 2001). O *I2C* tem como ponto forte a baixa complexidade de *hardware* aliado a um elegante conjunto de funcionalidades embutidas no próprio protocolo como: arbitragem, endereçamento do dispositivo escravo, confirmação de recebimento, detecção de colisões, entre outras que também serão discutidas neste capítulo.

O barramento é composto por apenas dois sinais: *Clock Serial* (*Serial Clock – SCL*) que transporta o *clock*, caracterizando o *I2C* como um protocolo síncrono; e o Dados Seriais (*Serial Data – SDA*), que é o sinal responsável por transportar os dados em ambas as direções, o que por sua vez caracteriza o protocolo como *half-duplex*. Ambas as linhas *SCL* e *SDA* são bidirecionais, logo os dispositivos *I2C* necessitam utilizar saídas do tipo dreno aberto, além de cada linha necessitar de um resistor de *pull-up* externo. Neste tipo de configuração é possível configurar uma ou mais unidades de controle como mestre e os demais dispositivos como escravos. Este protocolo de comunicação possui o nível lógico baixo como dominante, isto é, se mais de um dispositivo estiver controlando uma linha bidirecional ao mesmo tempo, o

dispositivo que estiver transmitindo o nível lógico baixo terá prioridade na rotina de controle. Outra característica das linhas bidirecionais utilizadas pelo protocolo *I2C* é a separação dos registradores de entrada e saída na porta dos dispositivos. Assim, um dispositivo mestre pode, por exemplo, escrever no registrador de saída da linha *SDA* o valor alto e verificar através do registrador de entrada que a linha *SDA*, na realidade, está em nível baixo, possivelmente sendo controlada por outro dispositivo conectado ao barramento. É como se todos os dispositivos presentes no barramento tivessem os registradores de saída de cada linha conectados a uma porta lógica *AND*. O protocolo *I2C* suporta múltiplos dispositivos mestre e múltiplos dispositivos escravos no mesmo barramento.

A comunicação entre dispositivos *I2C* é orientada a palavras de oito *bits*, para cada palavra de 8 *bits* transmitida um sinal de confirmação de recebimento, chamado *acknowledgement bit* (*ACK*), deve ser retornado pelo dispositivo receptor. Somente um dispositivo mestre pode iniciar uma comunicação, é sempre um dispositivo mestre que controla a linha de *clock* e cada dispositivo escravo possui um endereço físico único no barramento. O protocolo *I2C* também prevê alguns sinais de controle utilizados antes, durante e depois de qualquer transação. A Figura 7 indica a conexão física de dispositivo *I2C* à um barramento *I2C*.

Figura 7 – Conexão física de dispositivo *I2C* à um barramento *I2C*



Fonte: Leens (2009).

Antes de um dispositivo mestre iniciar uma comunicação, deve verificar se o barramento está disponível; quando as linhas *SCL* e *SDA* estão em nível lógico alto. Confirmando que o barramento está disponível, o mestre envia um sinal de controle chamado *start bit* que indica aos demais dispositivos que a palavra seguinte a ser transmitida contém um endereço de dispositivo escravo. Os endereços *I2C* possuem 7 *bits* de tamanho, o que permite endereçar pouco menos de 128 dispositivos escravos, pois alguns endereços são reservados para ações específicas. O *bit* menos significativo do *byte* (*Less Significant Bit - LSB*) que contém o endereço do dispositivo *I2C* sinaliza o tipo de transação, escrita ou leitura, da perspectiva do dispositivo mestre.

2.6 *INTERNET OF THINGS (IoT)*

A Internet das Coisas (*IoT – Internet of Things*) é uma tendência futura na comunicação de dados, e pode ser definida como uma infraestrutura de rede global dinâmica com capacidades de autoconfiguração baseadas em protocolos de comunicação padronizados e interoperáveis, onde "coisas" físicas e virtuais têm identidades, atributos físicos, e personalidades virtuais. A *IoT* utiliza interfaces inteligentes e são integrados perfeitamente na rede de informações, com diferentes protocolos para diferentes propósitos. O Quadro 1 indica alguns dos protocolos utilizados:

Quadro 1 – Protocolos de comunicação de dispositivos *IoT*

PROTOCOLO	**TRANSPORTE**	**MODELO DE INTERAÇÃO**	**DESCOBERTA**
CoAP	*UDP*	Requisição/Resposta	Sim
MQTT	*TCP/IP*	Publicação/Subscrição	Não
XMPP	*TCP/IP*	Ponto-a-ponto Troca de mensagem	Não

Fonte: Adaptado de Mohammed *et al.* (2016).

A seguir, serão descritos os protocolos citados anteriormente:

a) *Constrained Application Protocol* (*CoAP*):
   - projetado para ser leve e confiável, esta configuração destina-se à aplicações do tipo máquina-para-máquina (*Machine-to-machine – M2M*)*/IoT*. Utiliza o protocolo *User Datagram Protocol* (*UDP*) ou uma variante deste ao invés do protocolo *Transmission Control*

Protocol/Internet Protocol (*TCP/IP*). Pode ser considerado como uma versão reduzida do protocolo *Hyper Text Transfer Protocol* (*HTTP*) para dispositivos restritos, utilizando métodos básicos como *GET*, *POST* e *DELETE*;

b) Message Queue Telemetry Transport (MQTT):

– possui eventos de “pesos” leves e protocolo orientado a mensagem, permitindo que os dispositivos se comuniquem de forma assíncrona eficientemente através de redes restritas a sistemas remotos. Embora o *MQTT* e o *CoAP* tenham funções semelhantes, suas características são diferentes. A camada de transporte do *MQTT* é o *TCP*, enquanto do *CoAP* é *UDP*, e usa *Transport Layer Security* (*TLS*) para a segurança, enquanto o *CoAP* usa *Datagram Transport Layer Security* (*DTLS*). O modelo do *MQTT* é *Publish/Subscribe*, e para o *CoAP* é Transferência de Estado Representacional (*Request/Reply – REST*); não fornece a descoberta de recursos, enquanto o *CoAP* fornece, o *MQTT* usa métodos básicos como *Publish, Subscribe e Unsubscribe*;

c) Extensible Messaging and Presence Protocol (XMPP):

– é uma tecnologia aberta para comunicação em tempo real, utilizando a *Extensible Markup Language* (XML) como o formato base para troca de informações. Em essência, *Extensible Messaging and Presence Protocol* (*XMPP*) fornece uma maneira de enviar pequenos pedaços de *XML* de uma entidade para outra quase em tempo real.

A combinação da capacidade da próxima evolução da Internet (*IoT*) para coletar, transmitir, analisar, e distribuir dados em grande escala, com o processamento de informações, possibilita a aquisição de conhecimento para o desenvolvimento tecnológico futuro.

3 MATERIAIS E METODOLOGIA

Neste capítulo serão abordados os materiais e a metodologia empreendida para o desenvolvimento do presente trabalho.

3.1 METODOLOGIA

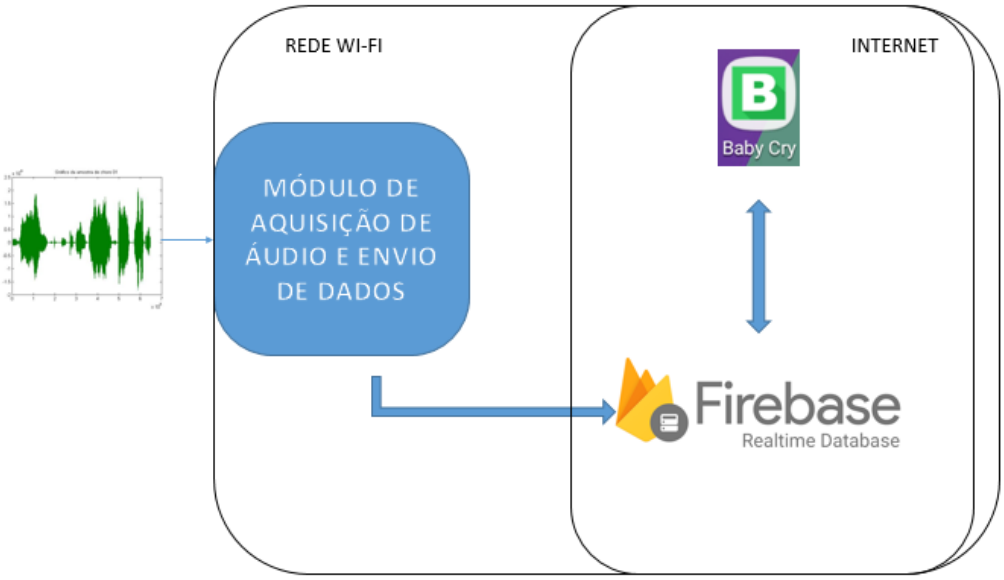
Em uma primeira etapa, áudios de choros de bebês serão coletados a partir de arquivos de domínio público disponíveis na *Internet*.

Os sinais de áudio serão capturados por um microfone com pré-amplificação do sinal, para que o mesmo esteja na faixa de intervalo de tensão da placa do *ESP-01*, que será responsável pelo processamento digital do sinal, através da detecção de picos de tensão.

Para a etapa de comunicação entre os dispositivos móveis e a placa de aquisição de áudio, em um primeiro momento, cogitou-se a comunicação direta entre ambos através da comunicação *Bluetooth*. Mas esta comunicação possui o fator limitador da distância que pode variar em um raio de até dez metros em ambientes abertos. Portanto, a forma escolhida para que a placa se comunique com o *smartphone* ou *tablet* foi através da rede *Wi-Fi*. A placa de comunicação *Wi-Fi ESP-01* pode trabalhar em modo *STA*, possibilitando que um dispositivo móvel se conecte diretamente a ela, porém, este modo impossibilita o acesso do *smartphone* ou *tablet* a outras redes *Wi-Fi*, limitando assim, as funcionalidades destes aparelhos. Para que o protótipo seja funcional, a forma de comunicação entre os dispositivos móveis e a placa de aquisição de áudio escolhida foi com o uso do módulo *ESP-01* configurado como *AP*.

Para que o protótipo alcance sua finalidade, a placa de conexão *wireless* será responsável pelo envio de dados obtidos a um banco de dados em tempo real conectado na *Internet*, que por sua vez, se encarregará de disponibilizar os dados ao *App Android*. Por fim, o *App* acionaria os dispositivos de atuação presentes no *smartphone* e/ou *tablet* dos usuários responsáveis pelos cuidados do bebê. A Figura 8 ilustra um fluxograma de funcionamento do protótipo:

Figura 8 – Fluxograma do funcionamento do protótipo



Fonte: Autoria própria (2019).

3.2 MATERIAIS

Inicialmente, fora proposto a utilização de uma placa *Arduíno Uno*, integrado à uma placa de comunicação *Wi-Fi*. No entanto, a capacidade de processamento do *Arduíno Uno* é menor se comparada com a placa de comunicação *Wi-Fi* ESP-01, que possui um processador interno e pode trabalhar independente ao *Arduíno*, em modo *stand alone*. Ou seja, é economicamente e computacionalmente viável retirar o *Arduíno* e utilizar apenas a placa *ESP8266 ESP-01*, já que esta pode ser programada com o uso do ambiente de programação do próprio *Arduíno*. Porém, o *ESP-01* possui a desvantagem de não possuir um pino acessível ao seu ADC, que realiza conversões do sinal de tensão interna, com a finalidade de mensurar a tensão no próprio circuito e esta placa também não possui uma saída *Universal Serial Bus* (*USB*) para comunicar-se com o computador, sendo necessário a utilização de um ADC externo e um módulo conversor *USB* para realizar a programação da placa. Para que a placa *ESP-01* trabalhe desconectada do computador, em um ambiente que deve ser monitorado, a mesma necessita de uma fonte de alimentação CC externa e por isso, um módulo de fonte para matriz de contatos foi selecionada para realizar esta alimentação de tensão.

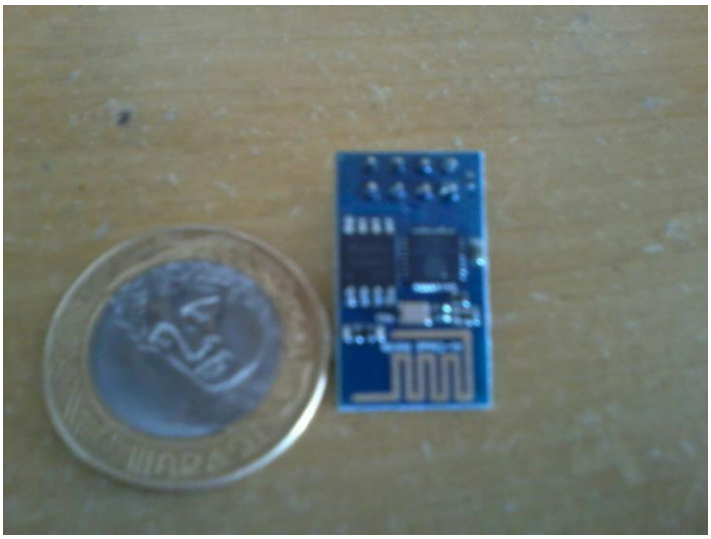
Portanto, para o desenvolvimento deste protótipo, os seguintes materiais foram utilizados: Placa de comunicação *WiFi ESP8266 ESP-01*, módulo ADC

PCF8591, módulo de fonte para matriz de contatos *YwRobot MB-V2*, placa *FTDI FT232RL* conversor *USB Serial*, e os *softwares*: *IDE Arduino*, *Android*, *Ionic*, *Apache Cordova*, *Pro Git*, *Node.js*, *EAGLE* e *Firebase*.

3.2.1 Módulo de Conexão *Wi-Fi ESP8266 ESP-01*

O módulo *ESP8266*, desenvolvido pela *Espressif Systems*, cujo modelo *ESP-01* é mostrado na Fotografia 1, possibilita a integração de um equipamento eletrônico à internet via *wireless* padrão 802.11 *B/G/N*, por meio de conexões *TCP/UDP* para o envio e recepção de dados, podendo ser configurado como *Access Point* (*AP*) e *Station* (*STA*).

Fotografia 1 – Módulo ESP8266 ESP-01 comparado com uma moeda



Fonte: Autoria própria (2016).

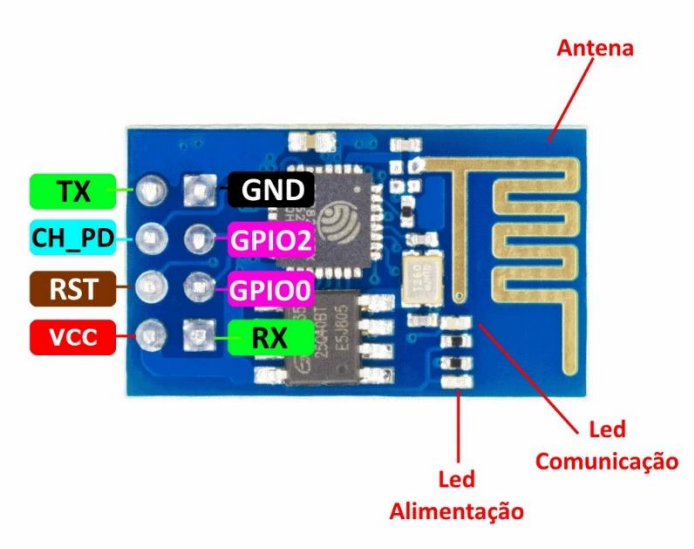
A comunicação deste módulo com o µC é realizada via interface serial, e através de dois pinos de entrada e saída de uso geral (*General Purpose Input Output - GPIO*), é possível configurar o módulo como uma aplicação *stand-alone*, ou seja, sem a necessidade de outro µC.

Possui barramentos *I2C*, *Serial Peripheral Interface* (*SPI*), *Universal Asynchronous Receiver/Transmitter* (*UART*), entrada ADC com 10 *bits* de resolução utilizada para medição de tensão interna, saída *Pulse With Modulation* (*PWM*) e sensor interno de temperatura, além de interface *Inter-IC Sound* (*I2S*) para aplicações de áudio. Sua *CPU* opera em 80MHz a 160MHz. Apresenta arquitetura *RISC* de 32

bits, memória *RAM* de 32 KB para instruções e 96 KB para dados, 64 KB de *ROM* e memória *Flash SPI Winbond* W25Q40BVNIG de 512 KB. O núcleo é baseado no *IP - Intelectual Property*; *Diamond Standard* LX106 da fabricante Tensilica.

A Figura 9 apresenta a pinagem deste módulo com as funções e especificações de cada pino.

Figura 9 – Pinos do módulo ESP8266 ESP-01



Fonte: Adaptado de Thomsen (2015).

Onde:

a) TX:
   – sinais transmitidos pelo módulo, deve ser conectado ao pino RX do *Arduino*;

b) RX:
   – sinais recebidos pelo módulo, deve ser conectado ao pino TX do *Arduino*;

c) VCC:
   – tensão de alimentação do módulo (de 3,3 V);

d) *GND*:
   – aterramento;

e) *RST*:
   – sinal de *Reset/Restart*, acionado em nível baixo;

f) CH_PD:

– sinal de habilitação do *chip* (*chip enable*), utilizado para gravação de *firmware* ou atualização (deve ser mantido em nível lógico ALTO para operação normal);

g) *GPIO0*:

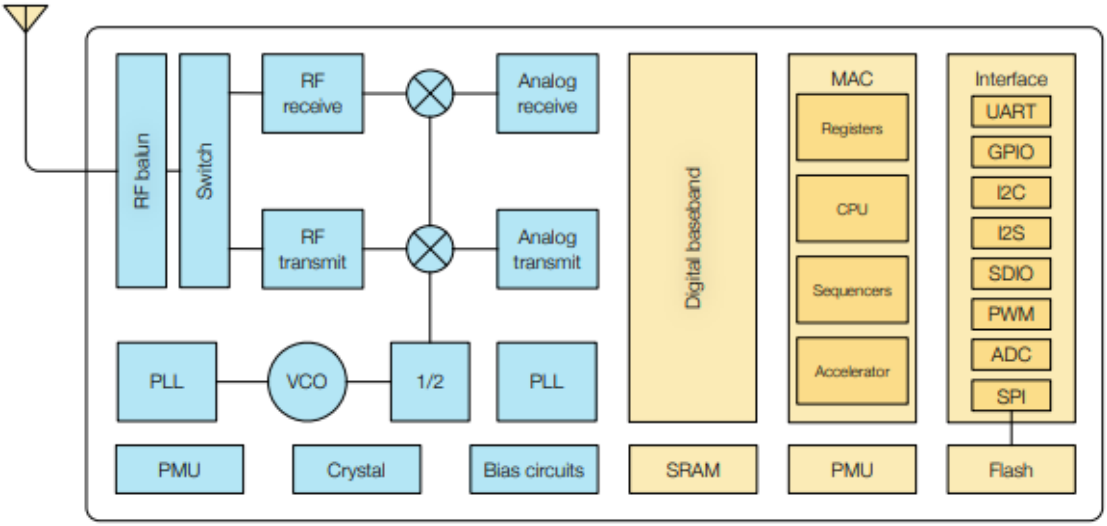
– pode ser configurado para controle pelo *firmware*, ou deve ser colocado em nível lógico baixo (*GND*) para modo *update*, ou em nível lógico alto para operação normal;

h) *GPIO2*:

– *I/O* que pode ser configurada para controle via *firmware*.

Um diagrama de blocos presente no µC presente no *ESP8266 ESP-01*, é indicado pela Figura 10:

Figura 10 – Diagrama funcional de blocos do µC presente no *ESP8266 ESP-01*



Fonte: ESP8266EX (2018).

O módulo *WiFi ESP8266 ESP-01* integra um processador *Tensilica L106 32-bit* Computador com um conjunto reduzido de instruções (*RISC*), que alcança um consumo de potência extremamente baixo e um *clock* máximo de 160MHz. As pilhas de *Real-Time Operating System* (RTOS) e *Wi-Fi* permitem que 80% da capacidade de processamento esteja disponível para programação e desenvolvimento de *Apps* pelo usuário. A *CPU* inclui as interfaces abaixo:

a) interfaces *RAM/ROM* programáveis (*iBus*), que pode ser conectado com o controlador de memória e também pode ser usado para acesar o *flash*;

b) interface de dados *RAM* (*dBus*), que pode ser conectado com o controlador de memória;

c) interface *AHB* que pode ser usada para acessar o registrador. (ESP8266EX, 2018).

O sistema-em-um-*chip* – *System-on-a-chip* (SoC) *WiFi ESP8266 ESP-01* integra controlador de memória e unidades de memória incluindo *Static Random Access Memory* (*SRAM*) e *ROM*. MCU pode acessar as unidades de memória através das interfaces *iBus*, *dBus* e *AHB*. Todas as unidades de memória podem ser acessadas a pedido, enquanto uma memória arbitrária decidirá a sequência de execução de acordo com o momento em que essas requisições são recebidas pelo processador. De acordo com a versão atual do *Kit* de Desenvolvimento de *Software* (*SDK*), o espaço *SRAM* disponível para os usuários é atribuído como se segue:

a) Tamanho da *RAM* menor que 50 kB, isto é, quando o *ESP8266 ESP-01* trabalha no modo *STA* e se conecta com o roteador, o espaço máximo programável acessível na seção pilha mais dados é cerca de 50kB.

O µC presente na placa *ESP-01*, utilizado no projeto, possui uma porta de entrada analógica com 10 *bits* de resolução, entretanto, esta é utilizada para calibração da tensão interna, sendo necessário a utilização de um conversor externo.

Pode ser usado nas seguintes aplicações: tomadas inteligentes, automação residencial, redes *mesh*, controle industrial *wireless*, babás eletrônicas, câmeras *IP*, redes de sensores sem-fio, dispositivos *wearable*, dispositivos com reconhecimento de localização via *WiFi*, tags de identificação para rastreamento e segurança e também para aplicações de sistemas de posição *WiFi* de *beacons*.

3.2.1.1 *Shields*

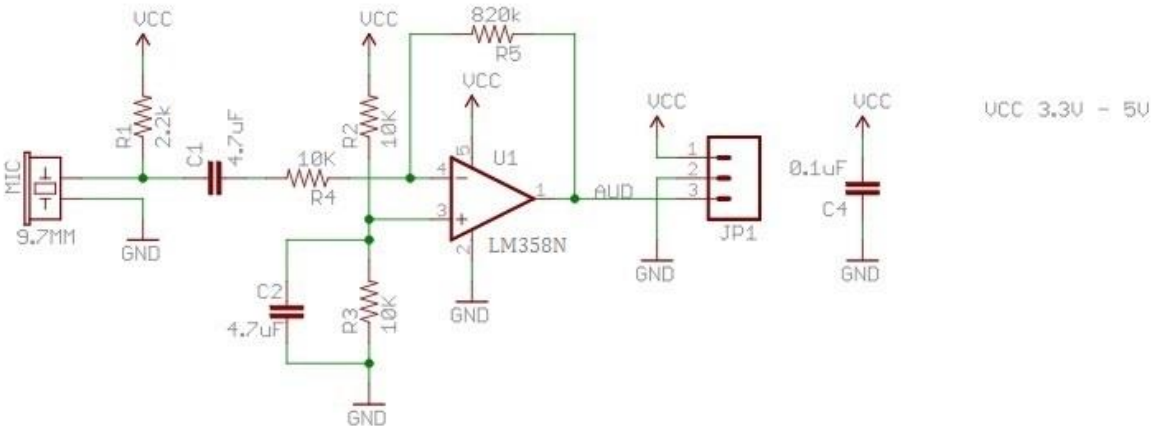
Uma das principais características do *Open Hardware*, um termo para artefatos tangíveis — máquinas, dispositivos ou outros objetos físicos — cujo projeto foi disponibilizado ao público de modo que qualquer um pode construir, modificar, distribuir e utilizar estes artefatos, é a compatibilidade com placas de *hardware* periféricas à placa principal do microcontrolador, denominadas *shields*, que possibilitam a expansão do sistema através de módulos contendo dispositivos como *displays* de cristal líquido (*LCD*), sensores de presença, receptores *Global Position*

System (*GPS*), que podem ser conectados diretamente ao *Arduino* através destas placas (MCROBERTS, 2011, p. 24).

3.2.1.1.1 Microfone

O modelo de microfone utilizado para a captura de áudio deste trabalho é semelhante ao do modelo *SparkFun Electret Microphone Breakout* (SEIDLE, 2016). O diagrama esquemático do circuito está representado na Figura 11. A diferença entre o circuito utilizado neste trabalho se encontra do tipo de circuito amplificador (LM344), substituído pelo circuito integrado LM358N.

Figura 11 – Diagrama esquemático do circuito eletrônico do *shield* microfone de eletreto



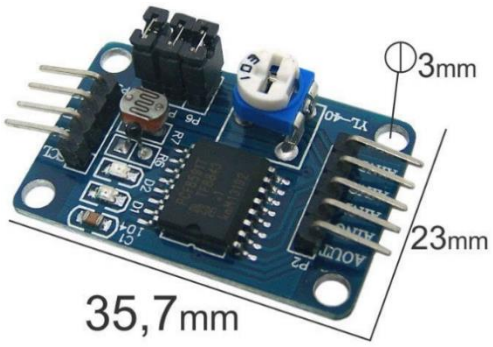
Fonte: Adaptado de Seidle (2016).

3.2.1.1.2 Módulo conversor AD/DA PCF8591

O Módulo *PCF8591* Conversor AD/DA, ilustrado pela Figura 12, é baseado no circuito eletrônico *PCF8591*, um *chip* capaz de converter sinais analógicos em digitais, além de digitais em analógicos. O Módulo *PCF8591* Conversor AD/DA é muito utilizado em projetos onde se faz necessária a utilização de diversos sensores na expansão de sinais analógicos.

Possui quatro pinos de entrada analógica e um pino de saída analógica com resolução de 8 *bits*. Opera na faixa de tensão de +2,5Vcc à +6Vcc e a comunicação é através do protocolo *I2C*.

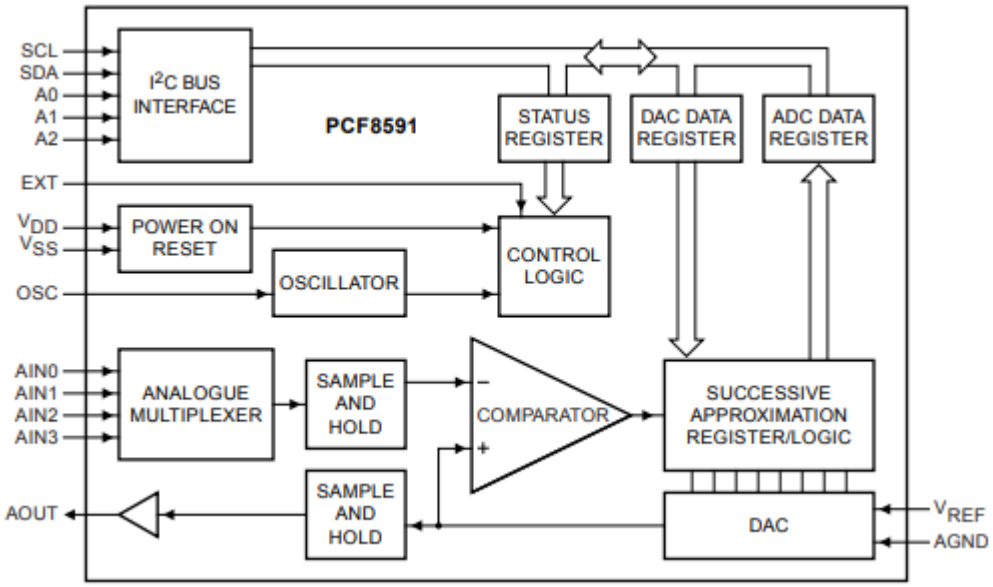
Figura 12 – Módulo AD/DA PCF8591



Fonte: Módulo (2019).

A Figura 13 ilustra o diagrama de blocos do módulo ADC *PCF85991*:

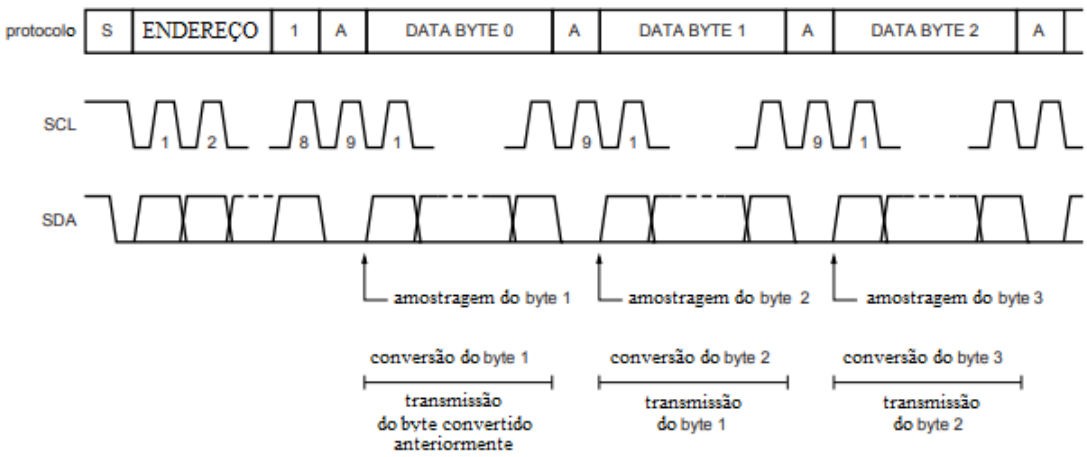
Figura 13 – Diagrama de blocos do *PCF8591*



Fonte: PCF8591 (2013).

Para a placa *PCF8591*, o ADC e um comparador de alto ganho são usados temporariamente durante o ciclo de conversão A/D. O ciclo de conversão A/D sempre é iniciado após o envio de um modo de leitura válido endereçado ao dispositivo *PCF8591*. O ciclo de conversão A/D é disparado na borda de descida do pulso de reconhecimento do *clock* e é executado durante a transmissão do resultado da conversão anterior, como ilustra a Figura 14:

Figura 14 – Sequência de conversão A/D



Fonte: Adaptado de PCF8591 (2013).

Uma vez que o ciclo de conversão é disparado, uma amostra de tensão de entrada do canal selecionado é armazenada no *chip* e é convertida para o código binário de 8 *bits*. Amostras recolhidas de entradas diferenciais são convertidas para o código de complemento de dois de 8 *bits*.

O resultado da conversão é armazenado no registrador de dados ADC e aguarda pela transmissão. Se o *flag* de autoincremento é habilitado, o próximo canal é selecionado. O primeiro *byte* transmitido no ciclo de leitura contém o resultado da conversão do ciclo de leitura anterior.

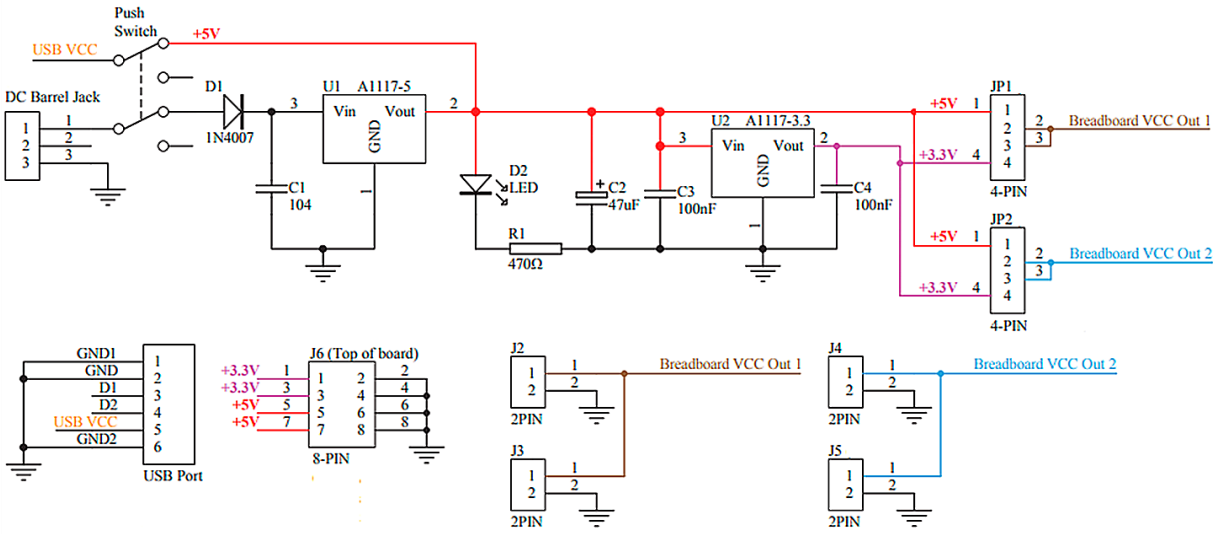
3.2.2 Módulo de Fonte de Tensão *YWROBOT MB-V2*

O módulo de fonte para matriz de contatos trata-se de um modelo de fonte não-ajustável, que é capaz de fornecer duas linhas de alimentação, de +5Vcc e +3,3Vcc por meio da configuração de *jumpers* na placa, e possui um conector *USB* para saída de tensão, assim como um conector de entrada para adaptador de energia. De acordo com as instruções de operação do módulo, ele deve ser alimentado com uma tensão de entrada entre +6,5Vcc e +12Vcc, e fornece em sua saída uma corrente máxima de 700mA, usando reguladores de tensão *AMS1117*. Tais reguladores são fabricados pela *Advanced Monolithic Systems*. No módulo existem dois desses reguladores: um para fornecer a tensão de +3,3Vcc e o outro para a linha de +5Vcc, ambos em encapsulamento *SOT-223*.

De acordo com o manual, esses circuitos integrados podem fornecer corrente de até 1A, suportam tensão de entrada de até +15Vcc, possui proteção interna contra curto-circuito, por meio de um circuito de limitação de potência. Também possui um circuito de proteção contra sobrecarga térmica, que desliga o regulador quando a temperatura de junção alcança cerca de 165ºC. Sua temperatura de operação máxima na junção é de 125ºC.

A Figura 15 indica o diagrama esquemático do circuito do módulo:

Figura 15 – Diagrama esquemático do circuito do módulo de tensão



Fonte: Adaptado de Addicore (2019).

3.2.3 Placa *FTDI FT232RL* Conversor *USB Serial*

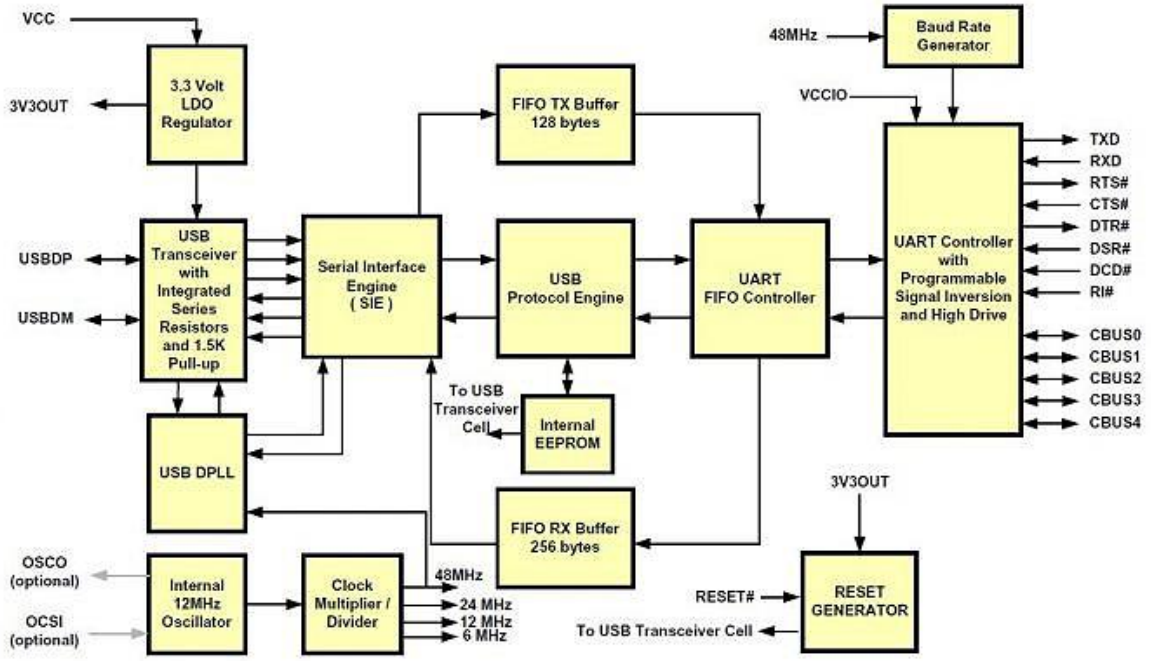
Esta placa *FTDI* é baseada no *chip FT232RL* que é um *chip* conversor *USB* – *UART* que usa seus pinos *TXD* e *RXD* para receber dados pela *USB* ou para transmitir dados para ela.

Este conversor pode ser alimentado com tensões de: +3,3Vcc a +5,25Vcc, sendo utilizada geralmente a tensão disponível pela interface *USB* (+5Vcc). Uma característica importante deste *chip* é um pino chamado *VCCIO*, que deve ser alimentado com a tensão a ser usada nos pinos de interface serial e no barramento *CBUS*. O pino *VCCIO* pode ser alimentado com tensões de: +1,8Vcc a +5,25Vcc. Em outras palavras, a tensão neste pino irá compatibilizar o nível de tensão nos pinos *TX* e *RX* com o microcontrolador, que geralmente utiliza +5Vcc ou +3,3Vcc. Para a comunicação com o nível de +5Vcc basta conectar *VCCIO* ao +5Vcc da *USB*. Já para

um nível de +3,3Vcc, o *FT232R* conta com um regulador interno do tipo *LDO* que disponibiliza essa tensão em um pino, o *3V3OUT*. Para outro nível de tensão, é necessário um regulador externo adicional.

A Figura 16 demonstra o diagrama de blocos da referida placa:

Figura 16 – Diagrama de blocos do FT232RL



Fonte: FT232R (2019).

O *FT232R* conta com um oscilador interno de 12 MHz, que necessita de uma alimentação no mínimo de +4Vcc, sendo que para valores abaixo deste é necessário um oscilador externo. A partir de um multiplicador/divisor interno, além dos 12 MHz, podem ser geradas as frequências de 6 MHz, 24 MHz e 48 MHz. Além disso, é possível configurar o uso em qualquer uma dessas frequências através de um pino, que pode ser usado para gerar o *clock* para um microcontrolador, eliminando a necessidade de um oscilador adicional.

A frequência de *clock* de 48 MHz gerada internamente pelo multiplicador é utilizada no bloco *USB DPLL* e no gerador de *Baud Rate* do *chip*. Isso possibilita ao *FT232R* ter uma transferência de dados de 183 *baud* a 3 *Mbaud*. A Figura 17 ilustra esta placa:

Figura 17 – Placa *FTDI FT232RL* conversor *USB* serial

Fonte: FTDI (2019).

Além dos pinos de transferência de dados, o *FT232R* possui os pinos para o controle de fluxo por *hardware*, um pino para *reset* que é ativo em nível baixo, um pino para habilitar o modo teste e um barramento chamado *CBUS* composto de cinco pinos que podem assumir várias funções.

O barramento *CBUS* pode ser considerado, para algumas aplicações, o bloco mais importante deste *chip*. Ele é composto de cinco pinos (*CBUS0* a *CBUS4*) que podem assumir diversas funções, bastando apenas configurar a *EEPROM* interna do *chip*.

Tais pinos podem assumir as seguintes funções:

a) TXDEN:
   – usado como habilitação de comunicação de dados no uso do protocolo RS485;

b) PWREN:
   – usado para habilitação da fonte de alimentação. Essa função faz do pino uma saída que fica em nível lógico “1” enquanto o dispositivo não é configurado pela *USB*, e após sua configuração passa para o nível “0”;

c) TXLED:
   – usado para acender um LED que indica a transmissão de dados pela *USB*. É uma função que usa o pino como saída e quando a *USB* transmite dados para o *chip*, envia pulsos em nível “0”;

d) RXLED:
   – usado para sinalização de transmissão de dados assim como TXLED, mas neste caso, pulsa quando o *chip* envia dados para a *USB*;

e) TX&RXLED:
   – uma junção de TXLED e RXLED, esta função pulsa em nível “0” o pino configurado quando o *chip* recebe ou envia dados para a *USB*;

f) SLEEP:
   – esta função configura um pino como saída em nível “0” quando a *USB* não está “aberta” por algum *software*, tipicamente usado em conversores *USB* – RS232 para desabilitar o *chip* de conversão de nível TTL- RS232;

g) CLK48, CLK24, CLK12 e CLK6:
   – são funções que configuram um dos pinos do CBUS para saída de *clock* de 48, 24, 12 ou 6 MHz;

h) CbitBangI/O:
   – com essa função é possível usar os pinos do bloco CBUS (em exceção do CBUS4) como entrada e saída. É necessário fazer a configuração dessa função na *EEPROM* interna do *chip*, e após isso, por comandos é possível realizar a leitura ou escrita nos pinos. Esta função permite monitorar sinais ou acionar cargas diretamente pelo computador através de uma interface de *software*. Outra possibilidade, é a comunicação serial, como *SPI* ou *I2C*, via *software*, e usar os pinos do barramento CBUS para controlar outros CIs, possibilitando a leitura de sensores (de temperatura, pressão, aceleração) por meio de um conversor analógico digital, aumentar a quantidade de portas através de um *chip* de expansão de *I/O* e gerar tensões analógicas através de conversores digital-analógico.

3.2.4 *Softwares*

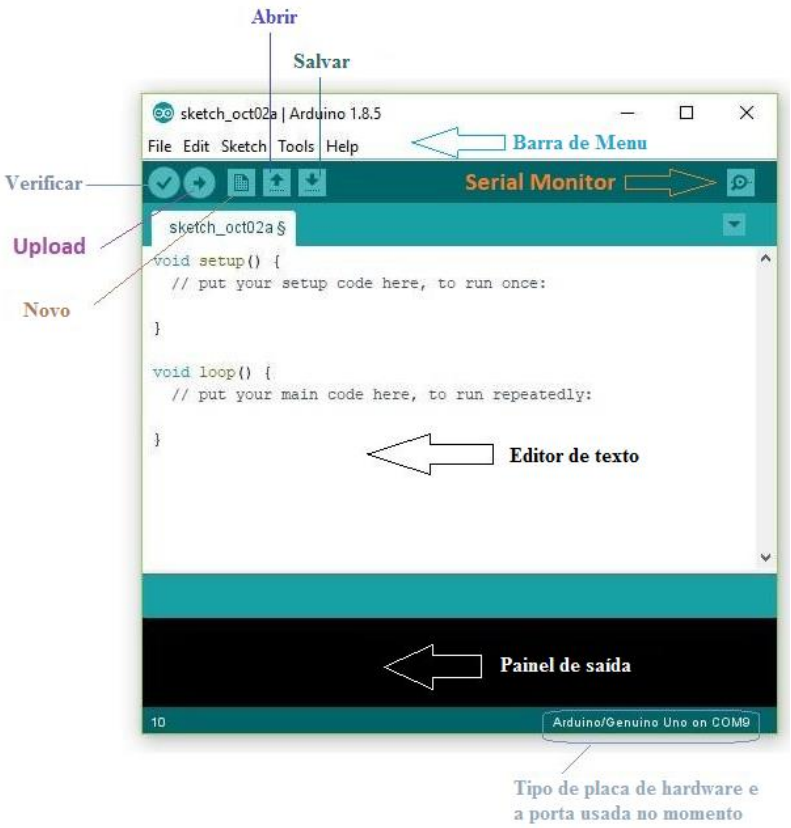
Nesta etapa, serão abordados os *software*s para a implementação do protótipo.

3.2.4.1 *IDE Arduino*

Composta por um Ambiente de Desenvolvimento Integrado (*Integrated Development Environment – IDE*), ilustrado através da Figura 18, utiliza uma linguagem de programação própria, similar à sintaxe da linguagem C, para a criação dos códigos fontes, também chamados *sketches*, que são enviados para a placa de *hardware*. (MCROBERTS, 2011, p. 24).

Segundo McRoberts (2011, p. 25), a plataforma de *software* também consiste do programa instalado previamente no *chip*, habilitando-o para a programação na *IDE*, identificado como *Arduino Bootloader*.

Figura 18 – *IDE Arduino*



Fonte: Adaptado de Aqeel (2018).

3.2.4.1.1 Bibliotecas Arduino

Uma biblioteca é um trecho de *software* que fornece funcionalidade específica a um programa, como por exemplo a capacidade de escrever em um *LCD* ou de controlar a posição de um servo-motor. O uso de uma biblioteca simplifica o desenvolvimento de aplicações, pois o código da biblioteca já está pronto, e só necessita ser incorporado ao programa em desenvolvimento para que suas funções possam ser acessadas e utilizadas pelo desenvolvedor. Assim, é possível estender o uso do *Arduino* incorporando bibliotecas específicas durante o desenvolvimento de um *sketch*.

Existem três tipos de bibliotecas utilizadas descritas a seguir:

a) biblioteca *Core*:
   - a biblioteca essencial, vem pré-instalada no *IDE* do *Arduino* e é imprescindível para o desenvolvimento de programas, desde os mais simples (como piscar um *LED*) até projetos complexos, como realizar automação de uma residência (em conjunto com outras bibliotecas). Desta forma, a programação do *Arduino* fica muito simplificada, pois o programador não necessita entender o código da biblioteca – basta saber como usá-la. Algumas funções comuns, fornecidas pela biblioteca core, são as funções *digitalRead*, *digitalWrite*, *Serial.begin* e *analogRead*, entre outras;

b) bibliotecas padrão:
   - são incluídas na instalação do *IDE* do *Arduino*, porém não são incluídas por padrão nos projetos criados, pois o *Arduino* possui recursos de memória limitados, e assim, essas bibliotecas somente são incluídas de forma explícita quando necessário. A inclusão de uma biblioteca padrão é feita por meio de uma declaração "#*include*" no início do código do *sketch*. São elas: *EEPROM*, *Ethernet*, *Firmata*, GSM, *LiquidCrystal*, SD, *Servo*, *SPI*, *SoftwareSerial*, *Stepper*, *TFT*, *WiFi* e *Wire*;

c) bibliotecas adicionais:
   - essas bibliotecas são disponibilizadas por desenvolvedores diversos que contribuem voluntariamente com o *software* para a plataforma, e não são distribuídas por padrão com o *IDE* do *Arduino*. Para usá-las, é necessário baixá-las e então efetuar sua instalação por meio do *IDE*. Oferecem funções adicionais a bibliotecas existentes ou novas funcionalidades não presentes em nenhuma biblioteca padrão, permitindo estender o uso do *Arduino* de forma praticamente ilimitada.

3.2.4.2 Sistema operacional *Android*

Com o surgimento e aumento da utilização dos *smartphones*, houve a necessidade da criação de um sistema operacional de código aberto, acessível à maioria dos usuários de dispositivos móveis, capaz de dinamizar e melhorar a utilização destes aparelhos. A *Blackberry* é a empresa pioneira em *smartphones*, e

desenvolveu um sistema operacional proprietário fechado para seus equipamentos, um sistema que facilitava tarefas como a leitura de e-mails, envio de mensagens, e agendamento de tarefas. Em 2005, a empresa *Android Inc*. foi criada, e em seguida incorporada pela *Google*, com o objetivo de lançar um sistema de código aberto, voltado para aparelhos celulares e *tablets* (DEITEL *et al.*, 2013, p. 4).

O *Android* foi criado via licença de código aberto de modo a facilitar sua instalação, utilização, e desenvolvimento dos respectivos *Apps*, utilizados para diversas finalidades; como jogos, leitura de notícias, acesso a redes sociais, entre outros. Deitel *et al*, (2013, p. 18 - 19) afirmam que tais *Apps* podem ser desenvolvidos com a linguagem de programação *Java* por um *kit* de desenvolvimento de *softwares* (*SDK*); através do *IDE Eclipse*, entre outros.

3.2.4.3 *Ionic*

Ionic é um *framework* de código aberto – *open source*, para o desenvolvimento de *Apps* híbridos: pequenos *websites* que rodam em um navegador de dispositivos móveis e utilizam *HyperText Markup Language* (*HTML5*). A Figura 10, abaixo indica o logotipo da plataforma. É necessário seu uso em conjunto com um *wrapper*, que consiste em um objeto de *software* que permite encapsular outros objetos de modo a alterar as suas interfaces ou comportamentos. Tipicamente os objetos *wrappers* mais conhecidos e utilizados encapsulam tipos primitivos, como *int* (inteiro), em instâncias de objetos. Neste caso o *wrapper Integer* permite criar um objeto do tipo *Integer* que contém um tipo primitivo *int*.

Um *App* híbrido é um *website* executado em um *App* nativo possibilitando a utilização de qualquer linguagem *HTML*, *Cascading Style Sheets* (*CSS*) e *Javascript*; para o desenvolvimento do *App* final.

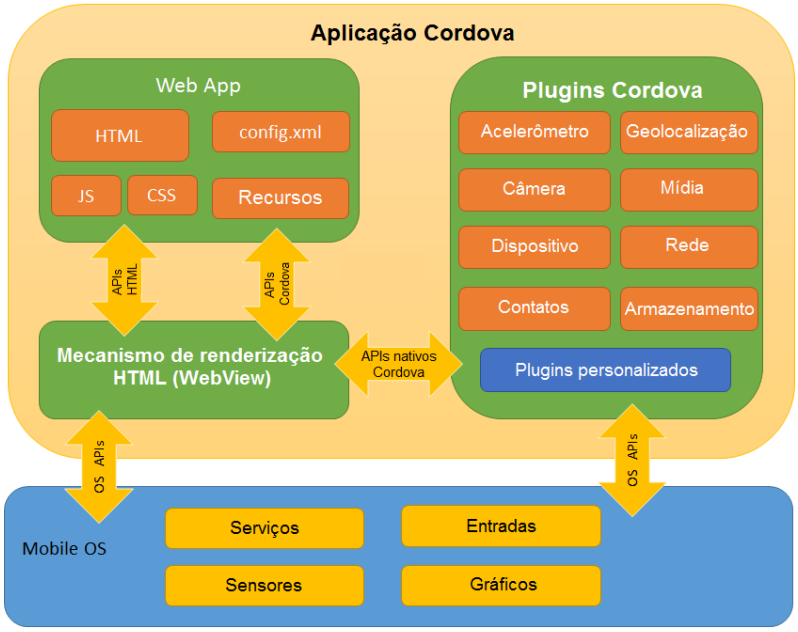
3.2.4.4 *Apache Cordova*

Para este trabalho foi necessário o uso de um *wrapper*, que é responsável pelo encapsulamento de objetos. O *Apache Cordova* possui esta funcionalidade, além do acesso aos sensores e periféricos presentes em um dispositivo móvel; como o acelerômetro, *GPS*, câmera, dados, entre outros; por meio do uso de *plugins*.

O *Apache Cordova* utiliza a ferramenta *WebView* para unir o *App* híbrido aos componentes nativos da aplicação. O código fica alocado no *Web App*, que fornece um importante arquivo, denominado *"config.xml"*, responsável por informações sobre o *App* e parâmetros específicos, influenciando o funcionamento do *App*.

Os *plugins* são parte integral do sistema *Cordova*, fornecendo a interface e comunicação entre cada componente nativo e ligações com as Interfaces de Programação de Aplicativos (*APIs*) do dispositivo padrão, como acesso aos componentes do dispositivo móvel: câmera, memória, *GPS*, e outros recursos. A Figura 19 exibe em diagrama de blocos a arquitetura presente neste *framework*.

Figura 19 – Arquitetura do *Apache Cordova*



Fonte: Adaptado de Overview (2016).

3.2.4.5 *Node.js*

Segundo Abernethy (2011, p. 1), *Node* é o interpretador *JavaScript* do *Google Chrome* para construir facilmente aplicações de rede rápidas e escaláveis. *Node.js* usa um modelo de *I/O* direcionado a evento não bloqueante, tornando-o leve e eficiente, ideal para aplicações em tempo real com troca intensa de dados através de dispositivos distribuídos.

Node utiliza o modelo de programação orientada a evento, em que uma conexão, assim como um dado recebido ou que deixou de ser recebido através da conexão, podem ser interpretados como eventos.

3.2.4.6 *Firebase*

Firebase é um *Backend as a Service* (*BaaS*) que oferece diversos serviços para facilitar o desenvolvimento de aplicações no lado do servidor. Com integração a várias plataformas como *Angular*, *JavaScript*, *Node.js*, *Android* e Sistema operacional *iPhone* (*IOS*), o *Firebase* ajuda desenvolvedores a focar no desenvolvimento *frontend mobile* e *web*.

O *Firebase Realtime Database* é um banco de dados hospedado na nuvem. Os dados são armazenados como notação de objetos *JavaScript* (*JavaScript Object Notation – JSON*) e sincronizados em tempo real com todos os clientes conectados. Quando *Apps* multiplataforma são criados com *SDKs* para *IOS*, *Android* e *JavaScript*, todos os clientes compartilham uma instância de *Realtime Database* e automaticamente recebem atualizações com os dados mais recentes.

4 DESENVOLVIMENTO

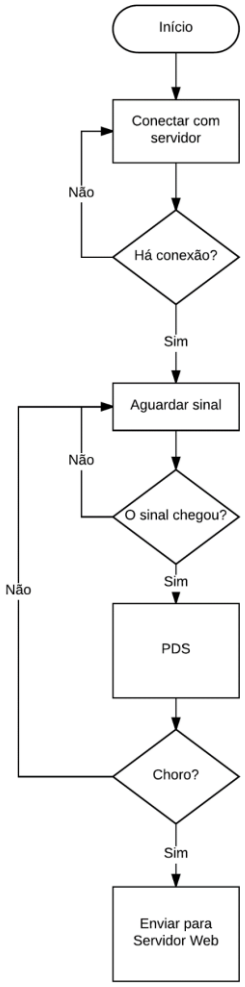
Este capítulo descreve cada etapa necessária para o desenvolvimento deste trabalho, sendo: entrada de áudio, processamento do sinal, comunicação de dados, banco de dados em tempo real, *App Android*, e o protótipo final da “babá eletrônica” para pessoas com deficiência auditiva.

Este protótipo possui três partes principais:

a) placa de captura e processamento de áudio e envio de dados;
b) banco de dados em tempo real;
c) *App Android*.

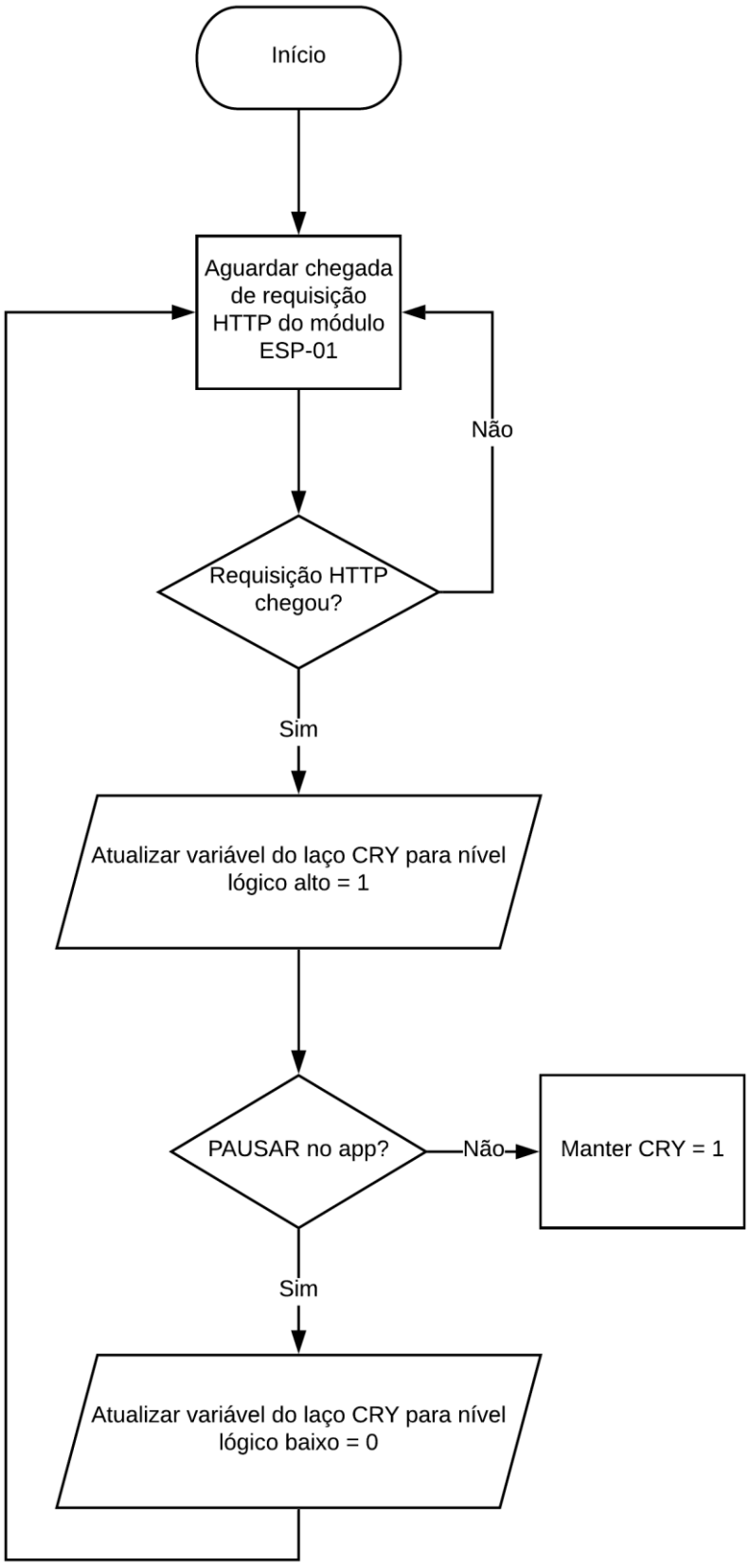
As Figuras 20-22 ilustram os fluxogramas de funcionamento deste projeto, com a respectiva explicação nos próximos subitens deste capítulo:

Figura 20 – Fluxograma de funcionamento do dispositivo de aquisição e processamento de sinal e envio de dados



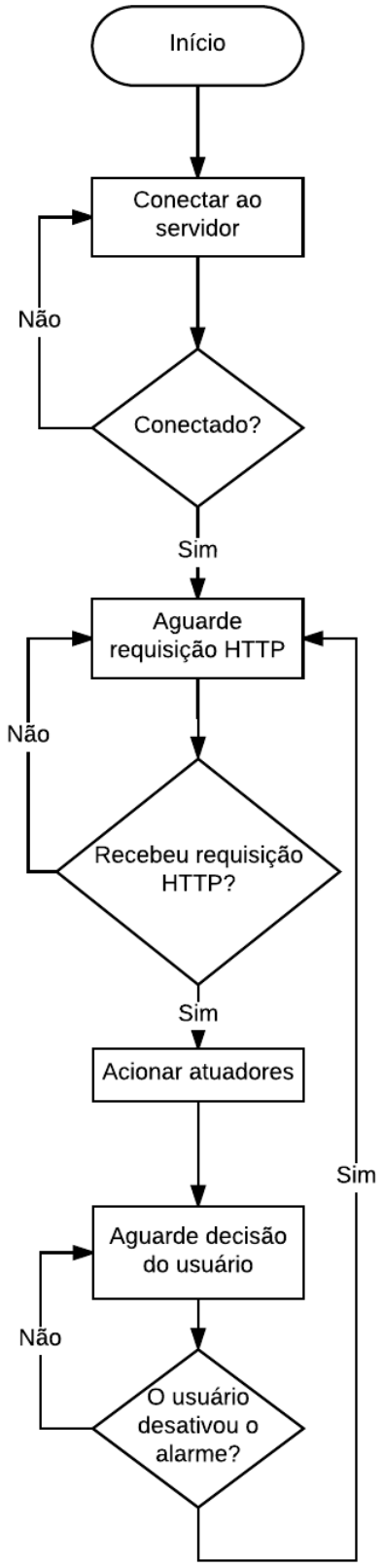
Fonte: Autoria própria (2016).

Figura 21 – Fluxograma de funcionamento do banco de dados



Fonte: Autoria própria (2019).

Figura 22 – Fluxograma de funcionamento do *App Android*



Fonte: Autoria própria (2019).

4.1 BANCO DE DADOS EM TEMPO REAL

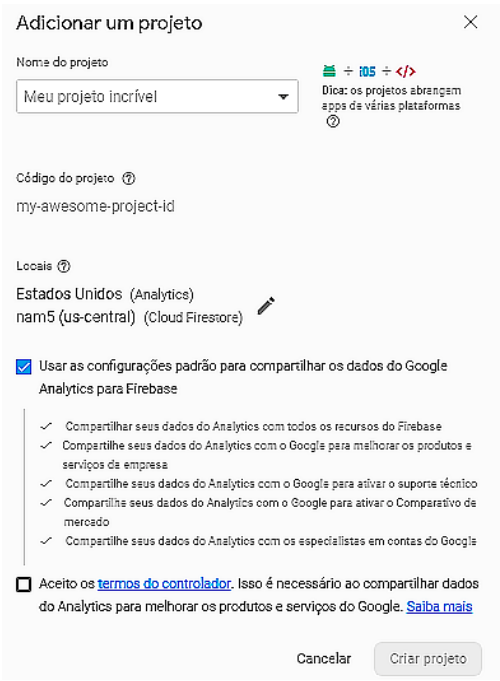
Com o uso do *Firebase*, o banco de dados construído em *Jscript*, é responsável pelo laço "*cry"* que possui uma única variável atualizável. Esta variável está configurada inicialmente em nível lógico baixo, aguardando a entrada de requisição do servidor *Web*. Ou seja, enquanto o sistema não detecta um sinal de intensidade sonora maior que a estabelecida, o banco de dados mantém o nível lógico baixo desta variável. Quando o módulo *ESP8266 ESP-01* executa a requisição *HTTP* do servidor *Web* ligado ao banco de dados, a variável presente no laço "*cry*" é atualizada para o nível lógico alto, disparando os atuadores do dispositivo pelo *App* conectado ao banco.

Para retornar ao estado de espera, a variável presente no laço "*cry*" deve retornar ao nível lógico baixo. Este procedimento é realizado somente no acionamento do botão "*STOP*" do *App*. Detalhes da programação estão presentes no APÊNDICE A.

As Figuras 23-25 ilustram os passos para a criação do banco de dados:

a) criação de uma conta do *Firebase*, adição de um novo projeto, seleção do local do servidor e local do *Cloud Firestore*;

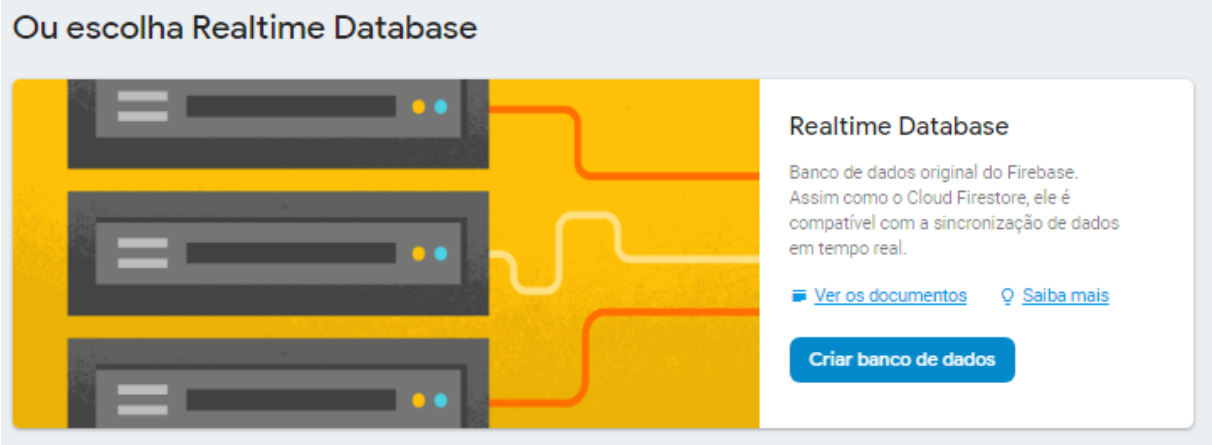
Figura 23 – Criação de um novo projeto do *Firebase*



Fonte: Autoria própria (2019).

b) criação do banco de dados em tempo real com *Realtime Database*;

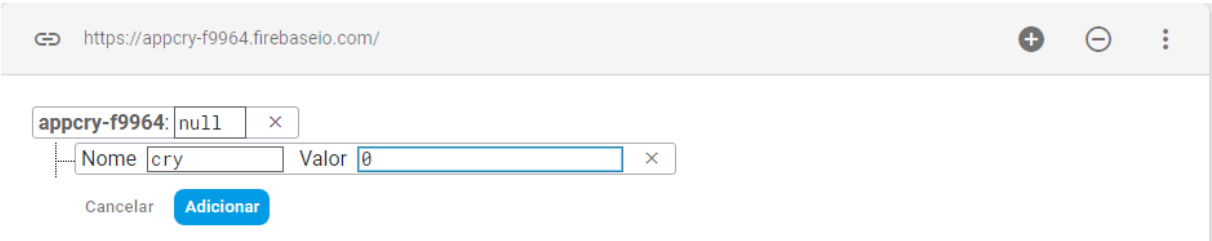
Figura 24 – Criação do banco de dados em tempo real



Fonte: Autoria própria (2019).

c) adição do laço “*cry*” com valor zerado.

Figura 25 – Criação do laço “*cry*”



Fonte: Autoria própria (2019).

4.2 *APP ANDROID BABY CRY*

Com o uso dos *softwares Node.Js*, *Angular*, *Cordova* e *Ionic*, foi desenvolvido um *App Android* conectado ao conjunto banco de dados/servidor *web* para acionamento dos atuadores do dispositivo, e para alterar o nível lógico da variável presente no laço “*cry*” do banco de dados, que comanda o retorno do sistema ao estado de espera de nova requisição mediante o acionamento do botão “*STOP*” da interface gráfica, desligando o dispositivo.

As Figuras 26 e 27 ilustram o ícone do *App* e sua interface, respectivamente:

Figura 26 – Ícone do *App*

Fonte: Autoria própria (2019).

Figura 27 – *App Baby Cry*

Fonte: Autoria própria (2019).

Detalhes da programação estão presentes no ANEXO B.

4.3 DESENVOLVIMENTO DO SISTEMA EMBARCADO

Nesta etapa serão abordadas as etapas para a construção do módulo de aquisição de áudio e envio de dados via conexão *WiFi*. A principal parte do sistema

embarcado é a placa *ESP8266 ESP-01*, que faz a análise do áudio e a conexão com o banco de dados do *Firebase*. Para que esta funcione, além das especificações de suprimento de energia, são necessárias preparações do ambiente de programação, bem como a atualização do *firmware* que se comunicará em conjunto com o *IDE* do *Arduino*.

4.3.1 *FTDI FT232RL*

Para que seja preparado, o *ESP-01* deve se comunicar com o computador, entretanto, a placa não possui saída *USB*. Para realizar a comunicação *UART*, o módulo de conversão *USB TTL* deve ser utilizado. O módulo *FTDI* possui saída de +3,3Vcc (tensão de alimentação do ESP-01) e para funcionar, os *drivers* disponíveis no site do fabricante devem ser instalados no Sistema Operacional.

4.3.2 Preparação da Placa *ESP-01*

Após os *drivers* da placa *FTDI232* serem instalados, é possível iniciar a comunicação entre o computador e a placa *ESP-01*, que deve ter sua versão de *firmware* atualizada para deixar o módulo mais estável e eliminar quaisquer inconsistências de versões anteriores.

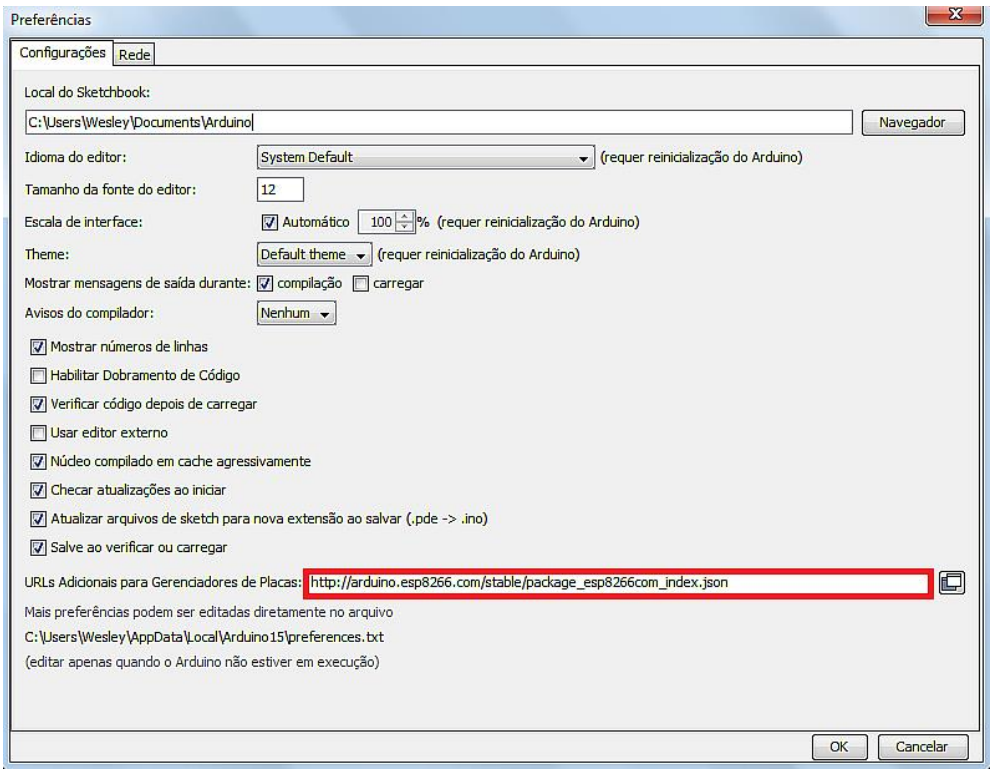
Para que seja realizado o *upgrade* do *firmware*, o módulo deve ser configurado em modo *flash*. Este modo é implementado com um pulso de nível lógico baixo no pino de *reset* enquanto a *GPIO0* está em nível lógico baixo.

Utilizando uma *protoboard*, *jumpers* de conexão e *push buttons* conectados aos pinos de *RST* e *GPIO0*, é possível atualizar o *firmware*. Para identificar a versão atual do *firmware*, o *software Termite* foi utilizado com a configuração de *Baud Rate* de 115200 *bps*, que é a taxa de comunicação serial padrão do ESP8266X. Em seguida, deve-se utilizar o *software ESP Flash Download Tool* em conjunto com um *firmware* mais atual do ESP8266 ESP-01 disponibilizado pela própria fabricante em seu site.

4.3.2.1 *IDE Arduino*

Para que o *ESP-01* receba as instruções que devem ser realizadas, a *IDE* do *Arduino* pode ser utilizada como compilador. Para isso, deve-se adicionar uma *Uniform Resource Locator* (*URL*) de gerenciamento de placas, nas preferências de configurações da *IDE*. A Figura 28 ilustra o procedimento da adição do link *http://arduino.esp8266.com/stable/package_esp8266com_index.json*.

Figura 28 – Ilustração da adição do gerenciador da placa *ESP-01*

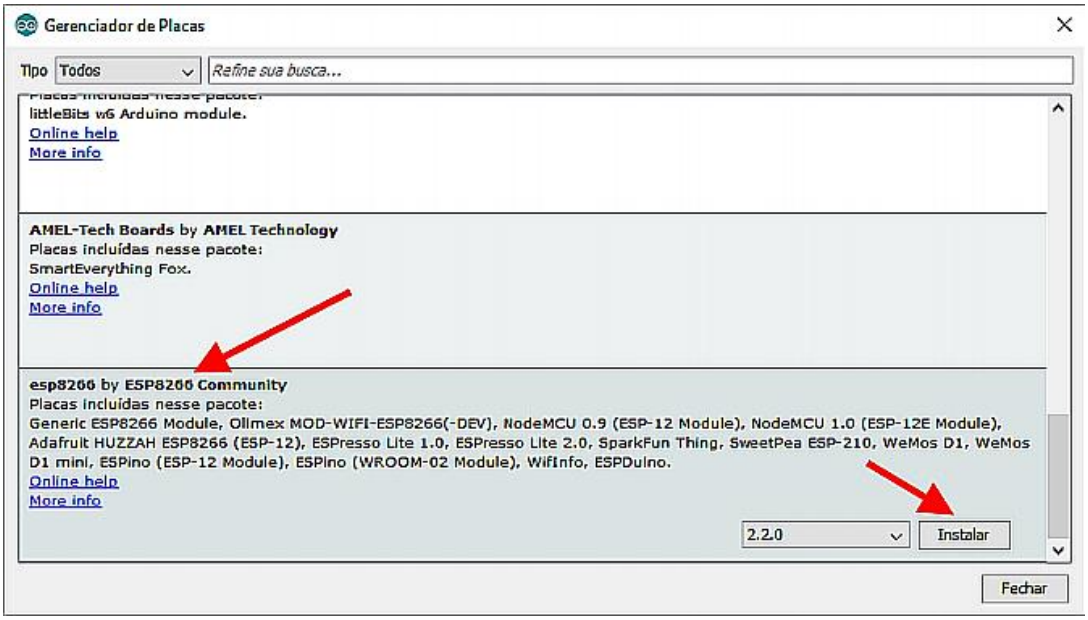


Fonte: Autoria própria (2019).

4.3.2.1.1 Bibliotecas

A *IDE* do *Arduino* permite a adição de bibliotecas que simplificam os *sketches*. Para que o *ESP-01* receba o *sketch*, se faz necessária a adição da sua biblioteca. Este trabalho também faz uso das bibliotecas adicionais *PCF8691* e *FirebaseESP8266*, disponíveis em um repositório *GIT*. Para adicionar uma biblioteca, é possível que o procedimento seja realizado através do próprio gerenciador de bibliotecas do *IDE* ou ainda, utilizando um arquivo "*.zip*", externo, como é o caso das bibliotecas encontradas no *GIT*. A Figura 29 exibe a tela deste procedimento.

Figura 29 – Adição de biblioteca

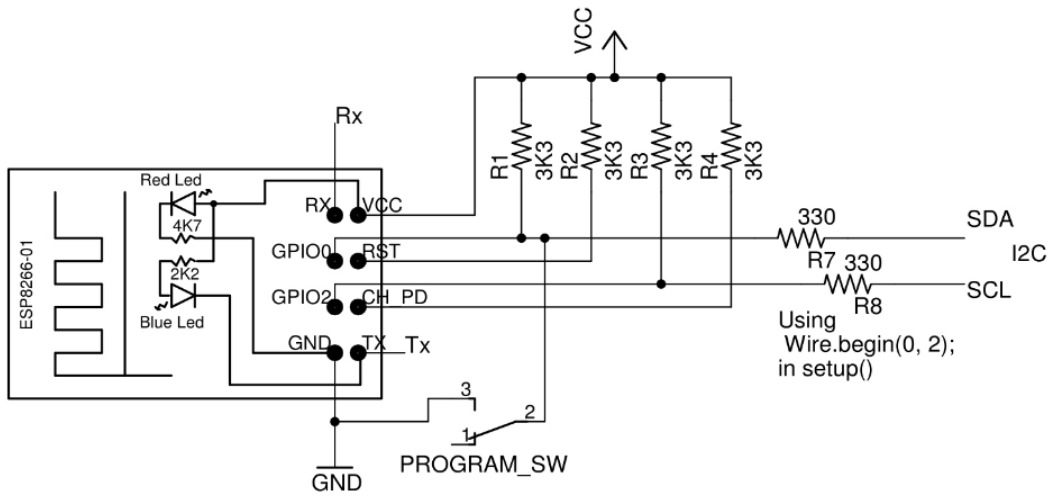


Fonte: Autoria própria (2019).

4.3.2.2 *I2C*

O *ESP-01* não possui um pino com saída ou entrada analógica de fácil acesso. As *GPIOs* são para sinais discretizados. Portanto, para que seja possível efetuar a leitura do sinal de áudio, que é analógico, se faz necessária a expansão de portas do módulo. Para isso, é possível utilizar a comunicação *I2C*, que pode ser implementada ao usar os pinos *GPIO0* e *GPIO2* e resistores *pull-up*, como ilustra a Figura 30:

Figura 30 – Esquemático de ligação para comunicação *I2C* do ESP-01



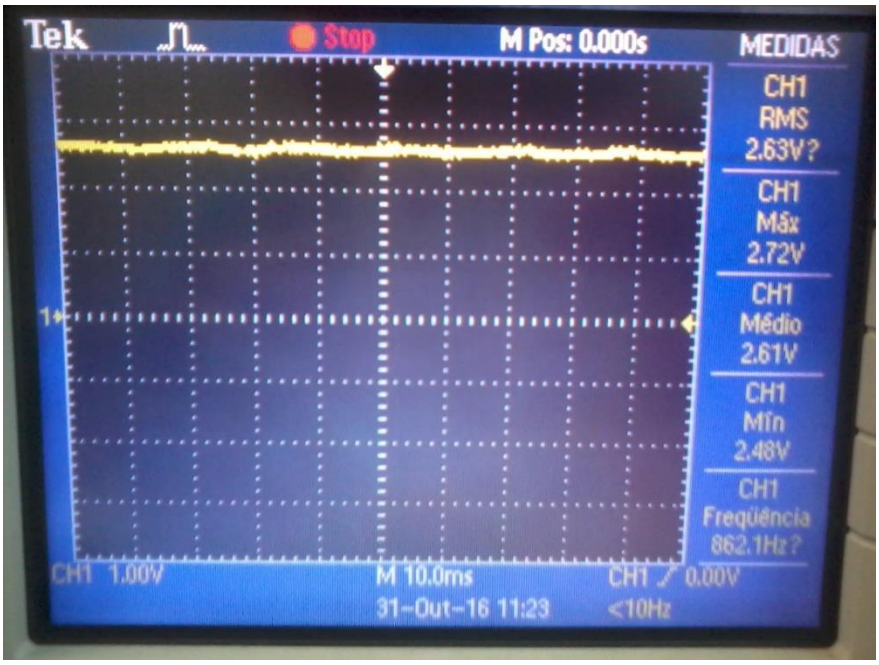
Fonte: Ford (2018).

As saídas *SDA* e *SCL* foram conectadas à placa *PCF8591*, e para que seja realizada a comunicação, se faz necessário informar o endereço utilizado. Para que este endereço seja identificado, um *sketch* para escanear o barramento *I2C* foi executado. Código disponível no ANEXO A.

4.3.3 Entrada de Áudio

O microfone escolhido para este trabalho visa atender à faixa de operação da placa ADC *PCF8591*, que varia entre +2,5 a +6Vcc. Este microfone possui uma amplificação para trabalhar na faixa de +3,3Vcc, o que possibilita operar as em conjunto com o *ESP-01*. A Fotografia 2 exibe o gráfico da saída da forma de onda capturada pelo microfone de eletreto após a amplificação da entrada de áudio em um ambiente silencioso.

Fotografia 2 – Forma de onda na saída da amplificação do sinal de áudio capturado pelo microfone de eletreto



Fonte: Autoria própria (2016).

Para alimentar o circuito do microfone, aplicou-se uma tensão de +3,3Vcc, presente em uma das saídas do módulo de fonte para matriz de contatos *YwRobot*.

Ao executar um áudio de choro, as tensões na saída do circuito variam em uma faixa de valores de 0 a +3,5Vcc.

4.3.4 Processamento do Sinal

Inicialmente, foi proposta a utilização de um método de processamento de sinal que filtrasse as frequências fundamentais presentes no choro de uma criança, facilitando a identificação do choro. Entretanto, partindo do pressuposto que este dispositivo é a versão inicial de um protótipo, a maneira encontrada para superar o problema da filtragem, tendo em vista que as opções testadas não forneceram resultados satisfatórios, foi a utilização da detecção de ruídos acima de uma amplitude sonora pré-definida.

A partir de medições das tensões máxima e mínima de pico, é possível calcular a tensão pico a pico e elaborar um comparativo entre a tensão indicada com a intensidade sonora, e por fim, estipular um limite que acione a requisição *HTTP*, disparando o alarme do *App* conectado ao banco de dados/servidor *web*.

4.3.5 Envio de Requisição

Após a detecção de intensidade sonora que ativa o sistema, é necessária a atualização da variável "*cry*", presente no banco de dados para que sejam acionados os atuadores do sistema. Inicialmente, uma requisição da *URL* onde o banco de dados está hospedado faria a alteração necessária. Logo, basta uma requisição através de uma chamada *HTTP* pelo *Client* por meio dos métodos básicos previstos pelo *CoAP*, tais como *GET*, *POST* e *DELETE* para atualizar a página do *host* do banco de dados. Porém, este método previsto pelo *CoAP* deve ser utilizado para o envio de informações ou requisição de uma informação presente na página do servidor *Web*.

Para otimizar o uso da memória do *ESP-01*, bem como corrigir a dificuldade de atualizar a página do *host*, foi decidido atualizar a variável "*cry*" diretamente no *Realtime Database*, e para isso, fora utilizada a biblioteca *FirebaseArduino*, disponível no *GIT*. No entanto, esta biblioteca apresentava desconexões frequentes com o servidor *web*, sendo necessária a atualização periódica de um arquivo da biblioteca para que fosse atualizado o certificado de segurança (*fingerprint*), da página do banco de dados, o que torna o sistema inoperante. Este problema foi contornado com o uso da biblioteca *FirebaseESP8266*, disponibilizada no gerenciador de bibliotecas do *IDE Arduino*. Os detalhes da programação do sistema embarcado estão presentes no APÊNDICE C.

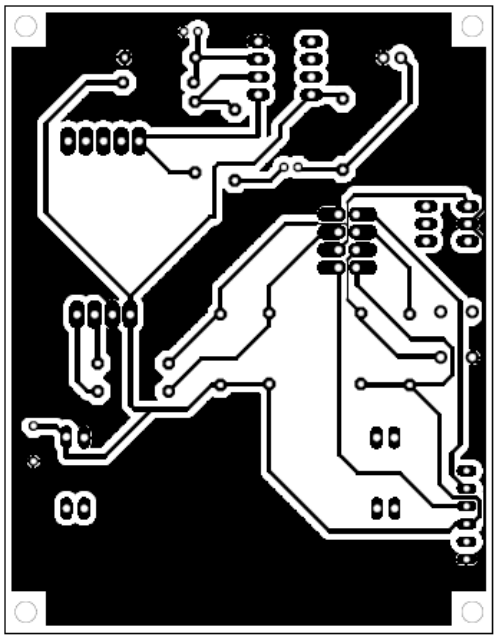
4.3.6 Placa PCB

Disponível para *Windows*, *MAC* e *Linux*, o *Autodesk EAGLE* é um *software* de automação do *design* de projetos eletrônicos (*EDA*), utilizado para projetos de placas de circuitos impressos (PCBs) e design de diagramas esquemáticos dos circuitos de sistemas embarcados. Possui uma ampla biblioteca de componentes eletrônicos para o desenvolvimento dos projetos. Para o desenvolvimento deste trabalho a versão 9.4.0 foi utilizada.

Após os testes realizados em *protoboard*, foi utilizado o *software Autodesk EAGLE* para construção do diagrama esquemático do sistema embarcado, bem como a criação do leiaute para confecção da PCB. O diagrama esquemático do circuito pode ser consultado no APÊNDICE D.

A Figura 31 ilustra a imagem invertida da face inferior da PCB, detalhando as rotas das trilhas:

Figura 31 – Visão das trilhas utilizadas

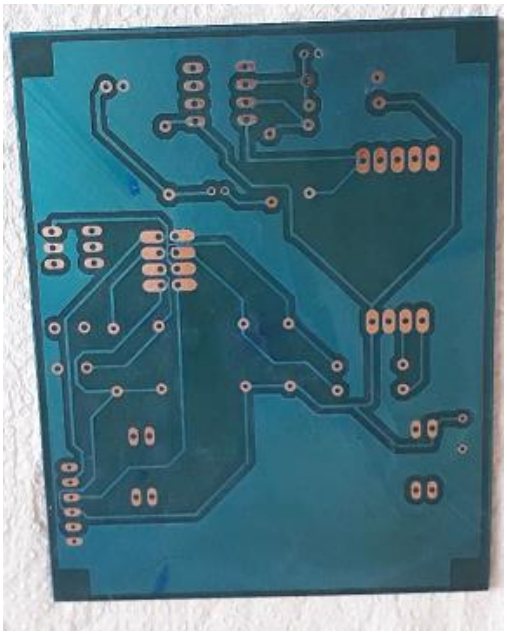


Fonte: Autoria própria (2019).

Para a confecção da placa, foi utilizado o método fotográfico, em que uma tinta fotossensível é aplicada à placa virgem e as trilhas são passadas para a placa através de um fotolito exposto a iluminação ultravioleta.

Ao final do processo, e corrosão da placa, a mesma finalizada é ilustrada na Figura 32:

Fotografia 3 – Vista inferior da placa



Fonte: Autoria própria (2019).

Após a confecção, foi realizada a perfuração e soldagem dos componentes. O sistema embarcado finalizado é ilustrado na Figura 33:

Fotografia 4 – Vista superior do sistema embarcado

Fonte: Autoria própria (2019).

5 RESULTADOS

Para discutir os resultados, se faz necessário o levantamento das etapas concluídas do protótipo:

O uso do banco de dados em tempo real *Firebase* foi criado com sucesso e a partir desta etapa, foi possível criar o *App "Baby Cry"* que se comunica com o banco de dados a todo instante.

O conceito inicial para o funcionamento do *App Android* "*Baby Cry*" estipulava disparar o sistema de vibração dos dispositivos móveis em conjunto com sinais luminosos dos mesmos ao atualizar o laço "*CRY"* do banco de dados e que a interrupção se daria após pressionar o botão de "*STOP"*. Foi possível constatar que o *App* pode ser aprimorado, com algumas modificações implementadas para melhorar o desempenho. Estas melhorias consistem na utilização de um *timer* que dispara a contagem de tempo no momento em que o banco de dados é atualizado, indicando o intervalo de tempo em que o sistema detectou a variação da intensidade sonora do ambiente; o descarte do acionamento de sinais luminosos, economizando a energia consumida pelo dispositivo móvel; e a inclusão de um alerta sonoro para cuidadores sem deficiência auditiva, possibilitando versatilidade ao protótipo.

Após a implementação destas melhorias, o *App* foi testado via teste realizado para requisição do endereço *URL* do servidor *web* a partir de um navegador de *Internet*, como o *Google Chrome*. O resultado obtido foi satisfatório para o ambiente de simulação: o dispositivo móvel teve seus atuadores disparados no momento em que a variável do banco de dados fora atualizada para o nível lógico alto. Finalizando o teste, no acionamento do botão "*STOP*" os alarmes do dispositivo foram desligados a partir da atualização da variável do banco de dados. Ou seja, o banco de dados em tempo real funcionou corretamente, bem como a integração com o *App* denominado *Android* "*Baby Cry*".

Optou-se por realizar a detecção de um sinal sonoro que ultrapassasse determinado valor de amplitude. Com o pressuposto de que os ambientes em que se faz o monitoramento encontra-se silencioso, é possível identificar a presença do choro da criança a partir do momento em que ela iniciar o choro. Entretanto, esta é uma abordagem que não exclui ruídos sonoros advindos de outras fontes, fazendo com que o sistema dispare com outras fontes sonoras.

Para estipular o nível de intensidade sonora a ser ultrapassado, foram realizados testes com o uso de um decibelímetro posicionado ao lado do sistema embarcado, que estava a uma distância de um metro da fonte de som. A tensão de referência foi ajustada em um nível de +2.00Vcc e foram realizados os testes de detecção do áudio com quatro arquivos de choro de domínio público, coletados da *Internet*, primeiro para o sistema embarcado com alimentação proveniente do módulo de conversão *USB-TTL* e depois para a fonte externa de alimentação de tensão contínua. Em seguida, o nível de tensão de referência foi reduzido em 0,01Vcc até que se fosse encontrada a tensão de referência que mais se adequa ao disparo do alarme. O sistema embarcado atualiza a variável do banco de dados para nível lógico alto todas as vezes que seu o nível de intensidade sonora for maior que o estipulado.

O Quadro 2 indica os resultados obtidos para tensão fornecida pelo módulo *FTDI*:

Quadro 2 – Valores de referência para sistema embarcado alimentado com o módulo conversor *USB* TTL

Nível de tensão (Vcc)	2.00	1.99	1.98	1.97
Intensidade sonora do ambiente silencioso (dBA)	32.5	32.4	38.7	34.3
Intensidade sonora no momento do disparo do sistema de alarme (dBA)	-	-	81.2	34.3
	-	-	80.0	34.3
	-	-	82.0	34.3
	-	-	82.1	34.3

Fonte: Autoria própria (2019).

É possível observar que para o valor de referência de +2.0Vcc não houve detecção do áudio. O mesmo ocorreu para o nível de tensão de +1.99Vcc. Ao estipular o valor para +1.98Vcc, o sistema conseguiu identificar o áudio para valores acima de 80 dBA e para o nível ajustado em +1.97Vcc, o sistema disparou em som ambiente, com 34.3 dBA. Sob esta perspectiva, é notável que o valor de ajuste de tensão que deve ser escolhido é de +1.98Vcc para o correto funcionamento do protótipo.

O Quadro 3 indica os resultados obtidos para tensão fornecida pela fonte de alimentação externa:

Quadro 3 – Valores de referência para sistema embarcado alimentado com o módulo de fonte de tensão externa

Nível de tensão (Vcc)	2.00	1.99	1.98	1.97
Intensidade sonora do ambiente silencioso (dBA)	33.2	34.4	35.1	38.4
Intensidade sonora no momento do disparo do sistema de alarme (dBA)	-	91.4	83.2	38.4
	-	-	81.3	38.4
	-	-	84.5	38.4
	-	-	80.7	38.4

Fonte: Autoria própria (2019).

É possível observar que para o valor de referência de +2.0Vcc não houve detecção do áudio. Para o nível de tensão de +1.99Vcc, houve a detecção para o primeiro áudio com uma intensidade de 91.4 dBA, que pode ser considerado elevado e não houve detecção para as outras amostras. Ao estipular o valor para +1.98Vcc, o sistema conseguiu identificar o áudio para valores acima de 80 dBA e para o nível ajustado em +1.97Vcc, o sistema disparou em som ambiente, com 38.4 dBA. É perceptível que o valor de ajuste de tensão que deve ser escolhido é de +1.98Vcc para o correto funcionamento do protótipo.

Em uma segunda etapa de testes, o *App* em conjunto com o banco de dados e com o sistema embarcado foram executados em conjunto e cumpriram satisfatoriamente suas funções de acordo com as simulações efetuadas anteriormente.

6 CONSIDERAÇÕES FINAIS

Este trabalho teve por objetivo a construção de um protótipo de uma “babá eletrônica” para pessoas com deficiência auditiva, utilizando os conceitos de desenho universal, equiparação de oportunidades e vida independente, proporcionando desta forma uma ferramenta auxiliadora para pessoas que cuidam de bebês.

Inicialmente era previsto o tratamento do sinal de áudio através de Transformadas Wavelet, mas devido à dificuldade na aplicação dos filtros na aquisição dos sinais, ou pelo µC, optou-se por realizar a abordagem de detecção de picos de intensidade sonora, apesar de não constituir o melhor método para a identificação de sinais sonoros característicos de choros de bebês, tendo em vista que permite a detecção de sons provenientes de outras fontes sonoras, disparando o alarme do sistema, prejudicando a eficácia do protótipo (portanto, melhorias futuras devem ser implementadas).

Dentre outras dificuldades na elaboração do projeto é possível citar a integração do sistema embarcado com a “nuvem”, pois o módulo de requisição da *URL* do servidor *web* não estava configurado com o *firmware* mais recente e o método empreendido inicialmente não era o apropriado para a integração entre o banco de dados, fazendo com que o sistema como um todo não estivesse interligado. Entretanto, a solução encontrada ao utilizar a biblioteca específica para o banco de dados pôde contribuir para a melhoria e integração do projeto na sua totalidade. O módulo é uma importante ferramenta, versátil e de custo acessível, que viabilizada o desenvolvimento e aumento do número de novos dispositivos *IoT*.

Entre as contribuições deste trabalho, é importante a aquisição de conhecimentos a respeito de novas tecnologias, como os dispositivos *IoT*, cuja expansão e desenvolvimento é uma tendência irreversível de mercado; a elaboração de *Apps*, ferramenta amplamente difundida e utilizada na sociedade atual, pela disponibilização crescente de equipamentos individuais com processamento embarcado, como *smartphones* e *tablets*. Bem como a utilização banco de dados e de ferramentas de projetos de sistemas embarcados.

Ao final deste projeto, deve-se levar em conta que o protótipo atendeu em partes a sua concepção, a um baixo custo de produção, indicando que é possível tornar a vida de pessoas com deficiência mais acessível, derrubando as barreiras que as limitam e trazendo mais qualidade de vida.

6.1 TRABALHOS FUTUROS

Como sugestões para trabalhos futuros, ou correlatos a este, é possível indicar:

a) a análise e implementação de filtros, digitais para diferentes plataformas de *hardware* ou filtros passivos na entrada do sinal para economia do custo computacional;
b) uma alternativa ao processamento do sinal por sistemas de reconhecimento de padrões, como transformadas Wavelet, para eliminar o maior número possível de disparos de alarme não provenientes do choro da criança;
c) projeto de uma fonte de alimentação própria para o sistema embarcado, tendo em vista que a alimentação proveniente do módulo de fonte para matriz de contatos gera excedente de componentes;
d) utilização de protocolos de segurança nas comunicações entre os dispositivos;
e) ampliação do banco de dados para trabalhar com mais dispositivos conectados porque neste projeto, o banco de dados está vinculado apenas ao protótipo. Para a viabilidade de um produto comercial, o banco de dados deve trabalhar com vários usuários ao mesmo tempo.

REFERÊNCIAS

ABERNETHY, M. **Just what Node.Js is?:** A ready-to-code server. [S.l.]: IBM Corporation, Mai. 2011. Disponivel em: https://kyulingcompany.files.wordpress.com/2012/06/os-nodejs-pdf.pdf. Acesso em: 4 set. 2016.

ADDICORE YwRobot Breadboard Power Supply Module Schematic. **GME**. V2. [S. l.: s.n.], Jan. 2019. Disponivel em: https://www.gme.cz/data/attachments/dsh.661-203.1.pdf. Acesso em: 25 maio 2019.

AQEEL, A. **Introduction to Arduino IDE**: A complete step by step tutorial on the Introduction to Arduino IDE. [Lahore]: The Engineering Projects, Out. 2018. Disponivel em: https://www.theengineeringprojects.com/2018/10/introduction-to-arduino-ide.html. Acesso em: 25 mai. 2019.

BRAGA, N. C. **Como funcionam os conversores A/D**: Parte 2 (ART224b). [S. l.]: Instituto NCB, Mai. 2010. Disponivel em: https://www.newtoncbraga.com.br/index.php/como-funciona/1509-conversores-ad-2. Acesso em: 25 maio 2019.

ÇENGEL, Y. A.; CIMBALA, J. M. **Mecânica dos fluidos:** Fundamentos e aplicações. Tradução de Katia Aparecida Roque e Mario Moro Fecchio. São Paulo: McGraw-Hill, 2007.

DEITEL, P. J. et al. **Android para programadores:** Uma abordagem baseada em aplicativos. Tradução de João Eduardo Nóbrega Tortello. Porto Alegre: Bookman, 2013.

ESP8266EX: Datasheet. **Espressif Systems**. Version 6.0. [S. l.]: Espressif, Nov. 2018. Disponivel em: https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf. Acesso em: 20 de maio de 2019.

FEDERAL BUREAU OF INESTIGATION. U.S. Department of Justice. **FBI 100:** The Lindbergh Kidnapping. [S. l.]: The FBI, Mar. 2008. Disponivel em: https://archives.fbi.gov/archives/news/stories/2008/march/lindbergh_030308. Acesso em: 12 nov. 2014.

FORD, M. **ESP8266-01 Pin Magic**: How to use the ESP8266-01 pins. [S. l.]: Forward Computing and Control Pty, Abr. 2018. Disponivel em: https://www.forward.com.au/pfod/ESP8266/GPIOpins/ESP8266_01_pin_magic.html. Acesso em: 21 de março de 2019.

FREITAS, F. A. **Babá eletrônica adaptada para surdos**. 2010. Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Controle e Automação) – Instituto de Engenharia de Sistemas e Tecnologias da Informação, Universidade Federal de Itajubá, Itajubá, 2010.

FT232R USB UART IC: Datasheet. **Future Technology Devices International**. Version 2.15. Glasgow: FTDI Chip, Abr. 2019. Disponivel em: https://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf. Acesso em: 10 de abril de 2019.

FTDI 232 USB to TTL. **NSK eletronics**, [S. l.: s. n.], 2019. Disponivel em: http://www.nskelectronics.in/index.php?route=product/product&product_id=904. Acesso em: 21 mai 2019.

GREUTHER, M. **Noguishi radio nurse**. Dearborn, MI: The Henry Ford, Mar. 2014. Disponivel em: https://www.thehenryford.org/explore/blog/noguchi-radio-nurse. Acesso em: 12 nov. 2014.

I2C_SCANNER. **Arduino**, [S. l.: s. n.], Nov. 2018. Disponivel em: https://playground.arduino.cc/Main/I2cScanner. Acesso em: 16 abr. 2019.

KALINSKY, D.; KALINSKY, R. **Introduction to I2C**. [Garden City, NY]: Embedded, Jul. 2001. Disponivel em: https://www.embedded.com/electronics-blogs/beginner-s-corner/4023816/Introduction-to-I2C. Acesso em: 25 maio 2019.

LAGASSE, L. L.; NEAL, A. R.; LESTER, B. M. Assessment of infant cry: acoustic cry analysis and parental perception. **Mental retardation and developmental disabilities research reviews**, Nova Iorque, n. 11, p. 83-93, abr. 2005. DOI: https://doi.org/10.1002/mrdd.20050. Disponível em : https://onlinelibrary.wiley.com/doi/pdf/10.1002/mrdd.20050. Acesso em: 20 mar. 2015.

LEENS, F. An introduction to I2C and SPI protocols. **IEEE Instrumentation and Measurement Magazine**, [S. l.], v. 12, n. 1, p. 8-13, Jan. 2009. DOI: https://doi.org/10.1109/MIM.2009.4762946. Disponível em: https://ieeexplore.ieee.org/document/4762946. Acesso em: 02 mar. 2019.

MCROBERTS, M. **Arduino básico**. Tradução de Rafael Zanolli. São Paulo: Novatec, 2011.

MICROCHIP. **ATmega48A/PA/88A/PA/168A/PA/328/P:** megaAVR® Data Sheet. [S.l.]: Microchip, Jan. 2018. Disponivel em: http://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061A.pdf. Acesso em: 15 mai. 2019.

MÓDULO PCF8591 Conversor AD/DA p/ Arduino. **Eletrodex eletrônica**, Barroso, MG: Eletrodex, 2019. Disponivel em: https://www.eletrodex.com.br/modulo-pcf8591-conversor-ad-da-p-arduino.html. Acesso em: 15 maio 2019.

MOHAMMED, A. et al. Gesture based IoT Light Control for Smart Clothing. *In*: **2016 IEEE International Conference on Emerging Technologies and Innovative Business Practices for the Transformation of Societies (EmergiTech)**, Balaclava, 3-6 ago. 2016. DOI: https://doi.org/10.1109/EmergiTech.2016.7737326. Disponivel em: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7737326. Acesso em: nov 2016.

OVERVIEW. **Cordova**. [S. l.: s. n.], c2016. Disponivel em: https://cordova.apache.org/docs/en/latest/guide/overview/index.html. Acesso em: 12 nov. 2016.

PARKER, S. **O livro do corpo humano**. Londres: DK, 2007.

PCF8591 8-bit A/D and D/A converter: Product data sheet. **NXP SEMICONDUCTORS**. Rev. 7. [S. l.]: NXP, Jun. 2013. Disponivel em: https://www.nxp.com/docs/en/data-sheet/PCF8591.pdf. Acesso em: 25 de maio de 2019.

PENIDO, É. C. C.; TRINDADE, R. S. **Microcontroladores**. Ouro Preto: IFMG, Jan. 2013. Disponivel em: https://docplayer.com.br/6952419-Microcontroladores-edilus-de-carvalho-castro-penido-ronaldo-silva-trindade.html. Acesso em: out 2016.

SASSAKI, R. K. **Inclusão:** construindo uma sociedade para todos. 7. ed. Rio de Janeiro: WVA, 2006.

SEARA, I. C.; NUNES, V. G.; LAZAROTTO-VOLCÃO, C. **Fonética e fonologia do português brasileiro**: 2º período. Florianópolis: LLV/CCE/UFSC, 2011. Disponivel em: http://ppglin.posgrad.ufsc.br/files/2013/04/Livro_Fonetica_e_Fonologia.pdf. Acesso em: 13 nov. 2014.

SECRETARIA DE DIREITOS HUMANOS DA PRESIDÊNCIA DA REPÚBLICA; SECRETARIA NACIONAL DE PROMOÇÃO DE DIREITOS DA PESSOA COM DEFICIÊNCIA; COORDENAÇÃO-GERAL DO SISTEMA DE INFORMAÇÕES SOBRE A PESSOA COM DEFICIÊNCIA. **Cartilha do censo 2010 – pessoas com deficiência: A deficiência no Brasil**. SDH-PR/SNPD. Brasília. 2012.

SEIDLE, N. **The electret microphone breakout**. v.20. [S. l.]: Sparkfun, Mar. 2016. Disponivel em: https://cdn.sparkfun.com/datasheets/BreakoutBoards/Electret_Microphone_Breakout_v20.pdf. Acesso em: set 2016.

SINGER, L. T.; ZESKIND, P. S. **Biobehavioral assessment of the infant**. Nova Iorque: The Guilford Press, 2001.

TESLA, N. My inventions. **Electrical experimenter**, Nova Iorque, v. VI, n. 1006, p. 696, fev. 1919. Disponivel em: https://www.americanradiohistory.com/Archive-Electrical-Experimenter/EE-1919-02.pdf. Acesso em: 02 jul. 2016.

THOMSEN, A. **Tutorial módulo wireless ESP8266 com Arduino**. [S.l.: s.n.], Jun. 2015. Disponivel em: http://blog.filipeflop.com/wireless/esp8266-arduino-tutorial.html. Acesso em: out 2016.

WEEKS, M. **Processamento digital de sinais utilizando Matlab® e Wavelets**. Rio de Janeiro: LTC, 2012.

ZANCO, W. D. S. **Microcontroladores PIC18 com Linguagem C - Uma abordagem prática e objetiva**. São Paulo: Érica, 2010.

APÊNDICE A – *Script* do banco de dados

```
<!DOCTYPE html>
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <title>Baby Cry</title>
  <meta name="description" content="">
  <meta name="viewport" content="width=device-width, initial-scale=1">
</head>
<body>
</body>

<!-- <script src="firebase/firebase.js"></script> -->
<script
src="https://www.gstatic.com/firebasejs/3.5.3/firebase.js"></script>
<script type="text/javascript">

function init() {
  // Initialize Firebase
  var config = {
    apiKey: "AIzaSyBf2DKgQbP1K5FB6_-p_IxcFNCzu6BTbfw",
    authDomain: "appcry-f9964.firebaseapp.com",
    databaseURL: "https://appcry-f9964.firebaseio.com",
    storageBucket: "appcry-f9964.appspot.com",
    messagingSenderId: "1012708929760"
  };
  firebase.initializeApp(config);

  firebase.database().ref("/").update({cry: 1});
}

window.onload = init;

</script>

</html>
```

APÊNDICE B – Programação do *App "Baby Cry"*

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="initial-scale=1, maximum-scale=1, user-
scalable=no, width=device-width">
    <title></title>

    <link rel="manifest" href="manifest.json">
    <link href="lib/ionic/css/ionic.css" rel="stylesheet">
    <link href="css/style.css" rel="stylesheet">

    <!-- ionic/angularjs js -->
    <script src="lib/ionic/js/ionic.bundle.js"></script>

    <!-- ngCordova -->
    <script src="lib/ngCordova/dist/ng-cordova.min.js"></script>

    <!-- cordova script (this will be a 404 during development) -->
    <script src="cordova.js"></script>

  </head>
  <body ng-app="cry">

    <ion-pane>
      <ion-nav-bar class="bar-balanced">
      </ion-nav-bar>
      <ion-nav-view></ion-nav-view>
    </ion-pane>

    <!-- your app's js -->
    <script src="js/module.cry.js"></script>
    <script src="js/routes.cry.js"></script>

    <!-- Utils and Libs -->
    <script src="lib/firebase/firebase.js"></script>
    <script src="lib/moment/min/moment.min.js"></script>
    <script src="lib/moment/min/locales.min.js"></script>
    <script src="lib/humanize-duration/humanize-duration.js"></script>
    <script src="lib/angular-timer/dist/angular-timer.min.js"></script>
    <script src="js/utils.js"></script>

    <!-- Main screen -->
    <script src="main/module.main.js"></script>
    <script src="main/controller.main.js"></script>
    <script src="main/services.main.js"></script>

  </body>
</html>

.center-div {
      position: absolute;
      margin: auto;
      top: 0;
      right: 0;
      bottom: 0;
      left: 0;
      width: 180px;
```

```
      height: 150px;
      border-radius: 3px;
}

.cry-button {
      font-size: 40px;
      height: 100px;
}


// CryApp

angular.module('cry', ['ionic', 'ngCordova', 'main'])

.run(function($ionicPlatform) {
  $ionicPlatform.ready(function() {

    if(window.cordova && window.cordova.plugins.Keyboard) {
      // Hide the accessory bar by default (remove this to show the
accessory bar above the keyboard
      // for form inputs)
      cordova.plugins.Keyboard.hideKeyboardAccessoryBar(true);

      // Don't remove this line unless you know what you are doing. It
stops the viewport
      // from snapping when text inputs are focused. Ionic handles this
internally for
      // a much nicer keyboard experience.
      cordova.plugins.Keyboard.disableScroll(true);
    }
    if(window.StatusBar) {
      StatusBar.styleDefault();
    }
  });
})


/* This is the routes to cry */

(function(){
  'use strict';

  angular
    .module('cry')
    .config(ApplicationConfig);

    //set dependencies for ApplicationConfig
    ApplicationConfig.$inject = ['$ionicConfigProvider', '$stateProvider',
'$urlRouterProvider'];

    function ApplicationConfig($ionicConfigProvider, $stateProvider,
$urlRouterProvider){
      //define routes
      $stateProvider
        .state('main', {
          url: '/main',
          templateUrl: 'main/main.html',
          controller: 'MainCtrl',
          controllerAs: 'main'
        });
```

```
      //set default state
      $urlRouterProvider.otherwise('/main');
    }

})();

/* Generic utility functions */

// Initialize Firebase
var config = {
      apiKey: "AIzaSyBf2DKgQbP1K5FB6_-p_IxcFNCzu6BTbfw",
      authDomain: "appcry-f9964.firebaseapp.com",
      databaseURL: "https://appcry-f9964.firebaseio.com",
      storageBucket: "appcry-f9964.appspot.com",
      messagingSenderId: "1012708929760"
};
firebase.initializeApp(config);

/* Este é o controlador da view "main" */

(function(){
  'use strict';

  angular
    .module('main')
    .controller('MainCtrl', MainCtrl);

    // Atribuindo as dependências de MainCtrl
    MainCtrl.$inject = ['$ionicPlatform', '$ionicPopup',
'$cordovaNativeAudio', '$scope'];

    // Controller da view "main"
    function MainCtrl($ionicPlatform, $ionicPopup, $cordovaNativeAudio,
$scope) {
      console.log('MainCtrl');
      /* jshint validthis: true */
      var vm = this;
      var id = 0;
      vm.cry = 0;

      // Assim que a plataforma estiver pronta ("ready"), fazer o pré-
carregamento
      // do arquivo de áudio do beep
      $ionicPlatform.ready(function() {

          $cordovaNativeAudio.preloadSimple('beep', 'audio/Censored_Beep-
Mastercard.mp3')
                  .then(function(msg) { console.log(msg); })
                  .catch(function(error) { console.error(error); });
      });

      // Executa a função vibrate automaticamente
      vibrate();

      /** FUNÇÕES **/

      // Adiciona um listener no valor remoto de "cry" e controla a
vibração e sons
      function vibrate() {
            firebase.database().ref("/").on('value', function(snapshot) {
```

```
          // Se o valor de "cry" for 1, o dispositivo deve vibrar e fazer
barulho
            if (snapshot.val().cry == 1) {
            // Atribui o valor 1 para o "cry" local
                  vm.cry = 1;

            // Iniciar o timer
                  $scope.$broadcast('timer-start');

            // Necessário armazenar o intervalo do beep para, futuramente,
interrompê-lo
            id = setInterval(function(){
                        $cordovaNativeAudio.play('beep')
                          .then(function(msg) { console.log(msg); })
                          .catch(function(error) { console.error(error);
});
                     }, 1000);

            // Ativar a vibração por 99999999 ms, ou aproximadamente 28
dias
            // Este plugin não funciona corretamente no iOS
                        navigator.vibrate(99999999);
                  }
           // Se o valor de "cry" for 0, é hora de parar a vibração e o
barulho
                  else if (snapshot.val().cry == 0) {
            // Atribui o valor 0 para o "cry" local
                        vm.cry = 0;
            // Para o timer na contagem atual
                     $scope.$broadcast('timer-stop');
            // Interrompe on intervalo do beep, silenciando o barulho
            clearInterval(id);
            // Interrompe a vibração
                        navigator.vibrate(0);
                  }
            });
      }

      // Função para desativação (ou parada, ou interrupção) das vibrações
e barulhos
      vm.deactivate = function() {
        // Se o "cry" local tiver valor 1, trocá-lo para 0 e atualizar o
"cry" remoto para refletir esta mudança
            if (vm.cry == 1) {
                  firebase.database().ref("/").update({cry: 0}).then(

            // Se obtiver sucesso, para o timer e muda o valor local de
"cry" para 0 (parando a vibração)
            function() {
              vm.cry = 0;
              $scope.$broadcast('timer-stop');
            },

            // Se falhar, mostra um popup com a mensagem de erro e
            // para a vibração (marcando "cry" como 0 e parando o timer)
            function() {
              $ionicPopup.alert({
                title: 'Erro',
                template: 'Falha na conexão com o banco de dados.',
                okType: 'button-balanced'
              });
```

```
              vm.cry = 0;
              $scope.$broadcast('timer-stop');
            }
          );
        }
      }
      }

})();


/* Este é o módulo da "main" */

(function(){
  'use strict';

  angular
    .module('main', ['timer']);

})();

/* Este é o serviço da "main" */

(function(){
  'use strict';

  angular
    .module('main')
    .service('Main', Main);

    // Atribuindo as dependências de Main
    Main.$inject = [];

    // Serviço da main
    function Main(){
      var self = this;
      };
})();

<ion-view view-title="Baby Cry">
      <ion-content scroll="false">
            <div class="center-div">
                  <button class="button cry-button button-block button-
balanced" ng-click="main.deactivate()">Pausar</button>
                  <h2 style="text-align: center;">Tempo</h2>
                  <h2 style="text-align: center;"><timer autostart="false"
interval="1000">{{hhours}} : {{mminutes}} : {{sseconds}}</timer></h2>
            </div>
      </ion-content>
</ion-view>
```

APÊNDICE C – *Firmware* do sistema embarcado

```
//Inclusão da biblioteca que faz integração com o banco de dados
#include <FirebaseESP8266.h>
//Inclusão da biblioteca que permite comunicação com o ESP-01
#include <ESP8266WiFi.h>
//Inclusão da biblioteca que permite a comunicação I2C
#include <Arduino.h>
//Inclusão da biblioteca que permite a comunicação com a placa ADC
#include <PCF8591.h>

//Definição do endereço web do banco de dados
#define FIREBASE_HOST "appcry-f9964.firebaseio.com"
//Definição da chave de acesso ao banco de dados
#define FIREBASE_AUTH "5j5DTYoYBRBsnsMzbXVhLVuD2NuarVf12TUWmN0m"
//Definição do nome da rede WiFi a se conectar - SUBSTITUIR REDE PELO SSID
#define WIFI_SSID "REDE"
//Definição da senha da rede WiFi – SUBSTITUIR PELA SENHA DA REDE WIFI
#define WIFI_PASSWORD "SENHA"
//Definição do endereço para comunicação I2C
#define PCF8591_I2C_ADDRESS 0x48


//Inicia a função de comunicação I2C entre o ESP-01 e PCF8591
PCF8591 pcf8591(PCF8591_I2C_ADDRESS, 0, 2);


//Declaração da variável da janela de samples
const int sampleWindow = 25;
//Declaração da variável de aquisição de áudio
unsigned int audio;


//Inicia a função do banco de dados
FirebaseData firebaseData;


//Início da setup do código, que é rodado apenas uma vez
void setup() {
//Atribuição da velocidade de comunicação serial
      Serial.begin(115200);
      Serial.println();


//Função que incia a conexão com rede WiFi
      WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
//Printa na tela Serial
      Serial.print("Connecting");
//Enquanto a conexão não é estabelecida é printado um ponto e dado um delay
de 300 milisegundos
      while (WiFi.status() != WL_CONNECTED){
            Serial.print(".");
            delay(300);
      }
//A partir do momento em que há conexão
      Serial.println();
//Printa na tela o endereço IP local
      Serial.print("connected with IP: ");
      Serial.print(WiFi.localIP());
//Inicia a função de comunicação para o Firebase
```

```
      Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
//Caso não se obtenha uma conexão com o servidor há uma nova tentativa
      Firebase.reconnectWiFi(true);
//O número de repetições de tentativas de reconexão é de 3
      Firebase.setMaxRetry(firebaseData, 3);
//Configura o número máximo de erro Queue
      Firebase.setMaxErrorQueue(firebaseData, 30);
//Habilita as requisições básicas GET, POST, SET, DELETE
      Firebase.enableClassicRequest(firebaseData, true);

//Inicia a sequencia de testes de conexões e funcionamento das funções
básicas
      String rules = "";
      bool readOk;

      Serial.println("\n-------------------------------------");
      Serial.println("Read Database Rules test...");

      if (Firebase.getRules(firebaseData)){
            readOk = true;
            Serial.println("PASSED");
            Serial.println("DATABASE RULES: ");
            Serial.println(firebaseData.jsonData());
            rules = firebaseData.jsonData();
            Serial.println("-------------------------------------");
            Serial.println();
      }
      else{
      Serial.println("FAILED");
      Serial.println("REASON: " + firebaseData.errorReason());
      Serial.println("-------------------------------------");
      Serial.println();
      }

      if (readOk){
      Serial.println("-------------------------------------");
      Serial.println("Write Database Rules test...");

            if (Firebase.setRules(firebaseData, rules)){
            Serial.println("PASSED");
            Serial.println("-------------------------------------");
            Serial.println();
            }
      else{
      Serial.println("FAILED");
      Serial.println("REASON: " + firebaseData.errorReason());
      Serial.println("-------------------------------------");
      Serial.println();}
      }
      delay(300);
//Inicia a comunicação com a placa ADC
      pcf8591.begin();
//Atribui um delay de 5 segundos
      delay(5000);
}

//Início do código principal, que é rodado repetidamente
void loop() {

//Atribuições de variáveis
      unsigned long start = millis();          //Início da janela de sample
```

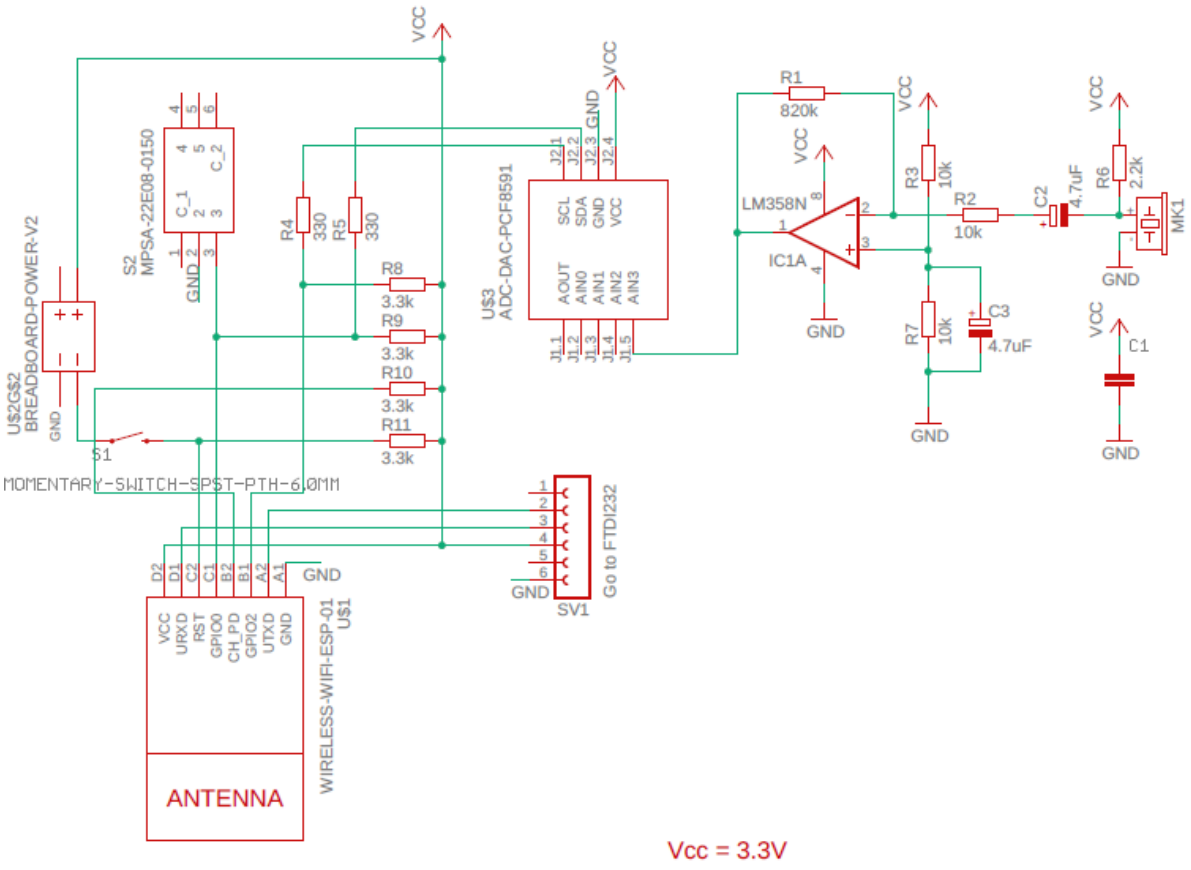
```
      unsigned int peakToPeak = 0;             //Nível de pico-a-pico
      unsigned int signalMax = 0;              //Nível máximo
      unsigned int signalMin = 255;            //Nível mínimo

//Coleta dados por 25 milisegundos
      while (millis() - start < sampleWindow){
//Define o valor da porta de entrada do sinal anaógico
      audio = pcf8591.analogRead(AIN3);
//O valor máximo do ADC de 8 bits, este loop incluirá todas as leituras
       if (audio < 255){
//Se o valor de entrada do audio for maior que zero
                  if (audio > signalMax){
//Salva o valor máximo de audio
                  signalMax = audio;
                  }
//Senão, se o valor de entrada do áudio foi menor que 255
            else if (audio < signalMin){
//Salva apenas os valores mínimos de audio
            signalMin = audio;
            }
        }
      }

// Variável pico-a-pico calcula a diferença entre o maior e o menor valor
de audio
peakToPeak = signalMax - signalMin;
// Declara a variável que realiza a conversão para Volts
float vpeak = (peakToPeak * 3.30) / 255;


//Printa os valores da tensão de pico
Serial.println(vpeak);
//Se a tensão for maior ou igual a 1.98[V]
      if(vpeak >=1.98){
//Altera o nó cry do Firebase para 1, disparando o alarme
      Firebase.setInt(firebaseData, "/cry",1);
      }
}
```

APÊNDICE D – Diagrama esquemático do circuito do sistema embarcado



ANEXO A – *Sketch* do *scanner* do endereço disponível para comunicação *I2C*

Este anexo apresenta o código para identificar o endereço disponível para comunicação *I2C*. (I2C_SCANNER, 2018).

```
#include <Wire.h>


void setup()
{
  Wire.begin();

  Serial.begin(9600);
  while (!Serial);
  Serial.println("\nI2C Scanner");
}

void loop()
{
  byte error, address;
  int nDevices;

  Serial.println("Scanning...");

  nDevices = 0;
  for(address = 1; address < 127; address++ )
  {
    // The i2c_scanner uses the return value of
    // the Write.endTransmisstion to see if
    // a device did acknowledge to the address.
    Wire.beginTransmission(address);
    error = Wire.endTransmission();

    if (error == 0)
    {
      Serial.print("I2C device found at address 0x");
      if (address<16)
        Serial.print("0");
      Serial.print(address,HEX);
      Serial.println("  !");

      nDevices++;
    }
    else if (error==4)
    {
      Serial.print("Unknown error at address 0x");
      if (address<16)
        Serial.print("0");
      Serial.println(address,HEX);
    }
  }
  if (nDevices == 0)
    Serial.println("No I2C devices found\n");
  else
    Serial.println("done\n");

  delay(5000);           // wait 5 seconds for next scan
}
```